VT-T00037-SSI
Doc-2012

xx(199614.1)

Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Laboratorio de Tecnologías de Información

Organización Semántica de Documentos Mediante Grafos

Tesis que presenta:

Erika Velázquez García

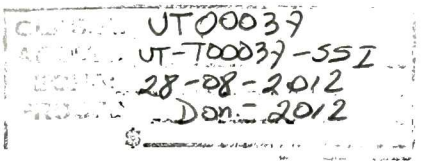
Para obtener el grado de:

Maestro en Ciencias en Computación

Director de la Tesis:
Dr. Iván López Arévalo

Cd. Victoria, Tamaulipas, México.

Diciembre, 2011



ID: 199479-2001

La tesis presentada por Erika Velázquez García fue aprobada por:

Dr. Víctor Jesús Sosa Sosa

Dr. Javier Rubio Loyola

Dr. Iván López Arévalo, Director

Cd. Victoria, Tamaulipas, México., 14 de Diciembre de 2011

A mi familia

Agradecimientos

- Quiero dedicar este trabajo a mi familia, por acompañarme y apoyarme en cada una de las decisiones que he tomado y ser siempre mi principal motivación para salir adelante.

- A mis amigos y compañero del CINVESTAV porque su amistad va más allá de un simple apoyo y compañía, porque cada uno de ustedes son la palabra de aliento o alegría que he necesitado.

- A mi asesor, porque muchas de estas páginas estarían vacías si no hubiera sido por su constante dedicación al ayudarme a concluir esta meta tan importante, gracias.

- Agradezco tambien al comité revisor de mi tesis, por sus correcciones y criticas constructivas hacia mi trabajo, que sin duda, influyeron en gran manera en la escritura de esta tesis.

- Al CINVESTAV por ser mi casa durante más de 2 años. Se que extrañare todas las largas y duras noches de trabajo. Me siento muy orgullosa de ser una de sus egresadas.

Muchas gracias a todos, desde el fondo de mi corazón.

Índice General

Índice General I

Índice de Figuras III

Índice de Tablas V

Índice de Algoritmos VII

Publicaciones IX

Resumen XI

Abstract XIII

1. Introducción 1
- 1.1. Antecedentes y motivación . . . 1
- 1.2. Planteamiento del problema . . . 5
  - 1.2.1. Hipótesis . . . 7
- 1.3. Objetivos generales y específicos del proyecto . . . 7
  - 1.3.1. Objetivo General . . . 7
  - 1.3.2. Objetivos Particulares . . . 7
- 1.4. Contribuciones y resultados esperados . . . 8

2. Estado del arte 9
- 2.1. Organización de documentos . . . 9
- 2.2. Sistemas de Recuperación de Información . . . 10
- 2.3. Medidas para la recuperación de información . . . 12
- 2.4. Herramientas existentes . . . 13
  - 2.4.1. Buscadores semánticos . . . 13
  - 2.4.2. Analizadores sintácticos para la extracción de texto . . . 16
  - 2.4.3. Etiquetadores . . . 18
- 2.5. Representación de texto . . . 18
  - 2.5.1. Latent Dirichlet Allocation . . . 20
- 2.6. Agrupamiento de texto . . . 23
  - 2.6.1. Clustering By Committee . . . 24
  - 2.6.2. Self-Organizing Map . . . 26
- 2.7. Cálculo de la similitud semántica . . . 28
- 2.8. Hipótesis Distribucional . . . 30
- 2.9. WordNet . . . 31
- 2.10. Trabajos relacionados . . . 32

2.11. Resumen 35

3. Metodología **37**
3.1. Metodología para la construcción del grafo semántico 38
3.1.1. Etapa I: Extracción de texto 40
3.1.2. Etapa II: Extracción de términos relevantes 41
3.1.2.1. Extracción de términos 41
3.1.2.2. Representación vectorial 44
3.1.2.3. Self-Organizing Map 47
3.1.2.4. Latent Dirichlet Allocation 49
3.1.3. Etapa III: Recuperación de significados 52
3.1.4. Etapa IV: Obtención de relaciones 54
3.1.5. Etapa V: Búsqueda en el meta-grafo 56
3.2. Estructura de datos para la representación de información 56
3.3. Búsqueda en el meta-grafo 57
3.3.1. Cálculo Local 58
3.3.2. Cálculo Global 61
3.4. Implementación en software 64
3.5. Resumen 67
3.6. Conclusiones 69

4. Pruebas y resultados **71**
4.1. Introducción 71
4.2. Casos de prueba 72
4.3. Validación experimental 74
4.3.1. Infraestructura de prueba 74
4.3.2. Salidas por etapas 74
4.4. Evaluación de la metodología 87
4.5. Resultados de los experimentos 90
4.6. Resumen 101
4.7. Conclusiones 102

5. Conclusiones y trabajo futuro **103**
5.1. Dificultades técnicas 104
5.2. Limitaciones 104
5.3. Contribuciones 105
5.4. Conclusiones 106
5.5. Trabajo futuro 109

A. Diagrama de clases de software de indexación y búsqueda **111**

Bibliografía **115**

Índice de Figuras

2.1. Diagrama de Ven que ejemplifica la relación de las métricas *Precision*(RR/R) y *recall*(RR/DR) 12
2.2. Representación gráfica del modelo de LDA. 22

3.1. Metodología propuesta para la construcción de un grafo semántico 39
3.2. Ejemplo de una matriz de salida de la red SOM 49
3.3. Ejemplo de recuperación de significados 54
3.4. Ejemplo de relaciones semánticas 55
3.5. Información que contienen los nodos del grafo. 57
3.6. Ejemplo de agrupación de términos por documento. 58
3.7. Ejemplo de representación gráfica de 5 relaciones que enlazan 4 documentos. 62
3.8. Herramienta utilizadas en los bloques que integran la solución de la representación gráfica. 66

4.1. Resumen y conclusiones del Documento 1. 75
4.2. Resumen y conclusiones del Documento 2. 76
4.3. Resumen y conclusiones del Documento 3. 77
4.4. Fragmento de la matriz de salida para el cálculo de *tf-idf* del Documento 2. 78
4.5. Salida de la red SOM para el Documento 2. 82
4.6. Clases que se generan como salida de la red SOM para el Documento 2. 83
4.7. Agrupación de relaciones por tema generados por LDA para el Documento 2. 84
4.8. Fragmento de términos derivados de la consulta a WordNet para los documentos 1, 2 y 3. 85
4.9. Ejemplo de la derivación de términos obtenidos de la consulta a WordNet para 4 palabras del Documento 2. 86
4.10. Grafo resultante de los términos derivados de los documentos 1, 2 y 3. 86
4.11. Gráfica comparativa de los valores de *precision* del Grafo obtenido y *Google Desktop* para el corpus de Springer 90
4.12. Gráfica comparativa de los valores de *recall* del Grafo obtenido y *Google Desktop* para el corpus de Springer 91
4.13. Gráfica comparativa de los valores de *precision* del Grafo obtenido y *SemanticVectors* para el corpus de Springer 92
4.14. Gráfica comparativa de los valores de *recall* del Grafo obtenido y *SemanticVectors* para el corpus de Springer 92
4.15. Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.6 96
4.16. Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.6 . . . 97

4.17. Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.7 98
4.18. Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.7 . 98
4.19. Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.8 99
4.20. Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.8 . 100

A.1. Diagrama de clases para proceso de Extracción de Texto. 111
A.2. Diagrama de clases para el Parser. 112
A.3. Diagrama de clases para el cálculo de *tf-idf*. 113
A.4. Diagrama de clases para JGibsLDA. 113
A.5. Diagrama de clases para la construcción y búsqueda del grafo. 114

Índice de Tablas

3.1. Matriz de pesos 46

4.1. Los grupos de noticias dividido por tema del corpus 20 Newsgroup 73
4.2. Documento 1, lista de verbos y sustantivos 79
4.3. Documento 2, lista de verbos y sustantivos 80
4.4. Documento 3, lista de verbos y sustantivos 81
4.5. Consultas realizadas sobre el corpus de Springer 88
4.6. Consultas realizadas sobre el corpus 20 Newsgroups para los temas `comp.os.ms-windows.misc` y `comp.sys.ibm.pc.hardware` 89
4.7. Consultas realizadas sobre el corpus 20 Newsgroups para el tema `sci.crypt` 89
4.8. Consultas realizadas sobre el corpus 20 Newsgroups para el tema `talk.religion.misc` 89
4.9. Tabla comparativa de los valores de *precision* y *recall* de la aplicación desarrollada y *Google Desktop* sobre el corpus de Springer 93
4.10. Tabla comparativa de los valores de *precision* de la aplicación desarrollada, *Google Desktop* y *SemanticVectors* para el corpus de *20 Newsgroups* 95
4.11. Tabla comparativa de los valores de *recall* para la aplicación desarrollada, *Google Desktop* y *SemanticVectors* para el corpus de *20 Newsgroups* 96
4.12. Tiempos de ejecución por cada etapa en la construcción del grafo 100

Índice de Algoritmos

1. Fase II de CBC 27
2. Algoritmo de muestreo de Gibbs para latent Dirichlet allocation 53

Publicaciones

Erika Velazquez-Garcia, Ivan Lopez-Arevalo, Victor Jesus Sosa-Sosa. *Representing Document Semantics by Means of Graphs*. 8th International Conference on Electrical Engineering, Computing Science and Automatic Control (2011). pages 963-968. Mérida, Yucatán, México. October 26-28, 2011.

Erika Velazquez Garcia, Ivan Lopez Arevalo, Victor Jesus Sosa Sosa. *Gestor Semántico de Documentos*. Libro electrónico: Avances en Informática y Sistemas Computacionales Tomo VI de la Colección Héctor García Molina. págs. 63-73.Universidad Juárez Autónoma de Tabasco. Cunduncán, Tabasco, México. Septiembre del 2011.

Resumen

Organización Semántica de Documentos Mediante Grafos

por

Erika Velázquez García

Maestro en Ciencias del Laboratorio de Tecnologías de Información
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2011
Dr. Iván López Arévalo, Director

Encontrar y acceder a la información relevante es esencial para un uso adecuado de la información y la adquisición de conocimiento, por ello este trabajo se enfoca en el problema de la búsqueda y recuperación de información. La búsqueda y recuperación de información generalmente se centra en la reducción inicial de un conjunto amplio de datos de entrada para pasarlo a un juego más concreto de información y hacer con ello un procesamiento extenso e intensivo necesario para la extracción y estructuración de conocimiento.

Para lograr una representación más compacta de documentos de texto no estructurados, en el idioma inglés, en este trabajo se propone utilizar como estructura de datos y medio de representación un grafo no dirigido. Para ello se describe un marco de trabajo para la representación y organización de documentos sin intervención humana, basado en la cooperación de diferentes algoritmos y técnicas de muestreo de texto que se utilizan para la extracción de características del conjunto de datos de entrada. También se incluye el desarrollo de un algoritmo para el cálculo de similitud de documentos, el cual se utiliza para la búsqueda de información. Dicho algoritmo toma en cuenta la consulta de entrada de un usuario y los diferentes significados que ésta puede tomar entre los documentos. Otro aspecto que considera este algoritmo son las diferentes temáticas a las que se pueden asociar los documentos, dando como resultado una búsqueda basada en semántica.

Un aspecto adicional que se cubre en este trabajo es la cantidad de documentos que se retornan al usuario, los cuales van en relación de los términos de la consulta y la cantidad de documentos recuperados, con esto se busca mejorar la precisión de la información retornada.

Con base en los experimentos realizados se presenta el análisis y discusión de los resultados con los que se ha podido determinar las ventajas y observaciones de los resultados y estadísticas de la aplicación desarrollada para probar el método propuesto. También se describen las características de los grupos de datos y los escenarios de prueba así como las métricas de evaluación aplicadas para medir el desempeño de la propuesta.

Abstract

Document Organization by Using Semantic Graphs

by

Erika Velázquez García

Master of Science from the Information Technology Laboratory
Research Center for Advanced Study from the National Polytechnic Institute, 2011
Dr. Iván López Arévalo, Advisor

Finding and accessing relevant information is essential for proper use of information and knowledge acquisition, so this work focuses on the problem of searching and retrieving information. The search and retrieval of information usually focuses on the initial reduction from a wide range of input data to a more specific. With this information an extensive and intensive processing is required for extraction and structuring of knowledge.

To achieve a more compact representation of unstructured documents in English, this thesis proposes to use as a data structure and representation through an undirected graph. We describe a framework for the representation and organization of documents without human intervention, based on the cooperation of different algorithms and text sampling techniques used for feature extraction of an input dataset. Also is included development of an algorithm for calculating the similarity of documents, which is used to search information. This algorithm takes into account the input query from a user and the different meanings that it can take between documents. Another aspect considered by this algorithm are the different topics to which they can associate documents, resulting in a semantic-based search.

An additional aspect covered in this thesis is the number of documents that are returned to the user, which are related to the query terms and the number of retrieved documents. The goal is to improve the accuracy of the information returned

.

Based on the experiments, an analysis and discussion of the results is presented, which is able to determine the advantages and observations of results and statistics of the developed prototype to test the proposed method. It also describes the characteristics of groups of data and test scenarios and evaluation metrics applied to measure the performance of the proposal.

1

Introducción

En este capítulo se describen brevemente el contexto, objetivos y contribuciones de este trabajo de tesis. Asimismo se describe la estructura del documento.

1.1 Antecedentes y motivación

Los buscadores de documentos actuales como el explorador de Windows, o los comandos "find" y "grep" de Linux basan su búsqueda por nombre de archivo o contenido, pero éstas son búsquedas sintácticas y léxicas, lo que significa que la palabra o parte de la palabra a buscar debe coincidir de forma exacta con los nombres de los documentos o contenidos.

En las tareas de búsqueda de información en archivos, el primer instinto de un usuario común es intentar escribir con sus propias palabras lo que desea buscar, en vez del conjunto exacto de palabras que se encuentran en el texto. Un problema adicional es la enorme cantidad de documentos disponibles como fuentes de información que almacenamos en diferentes directorios y subdirectorios en un disco duro. Es muy poco probable que un usuario común tenga en mente la totalidad de la

organización de todos los subdirectorios y lo que contiene cada uno. Aunque toda esta información se encuentre disponible para nosotros es muy difícil encontrar la información exacta que realmente esperamos como resultado de una búsqueda. Esto es muy notable sobre todo en situaciones en las que un usuario tiene que buscar entre grandes cantidades de archivos o dispone de poco tiempo para hacerlo. Hay que considerar también que esta información se encuentra en diferentes formatos y diferentes formas de organización. Este problema tiene como consecuencia el que se descarten ciertos resultados importantes en una búsqueda, porque no le es posible al buscador "interpretar" el contenido de dichos documentos. Es en este punto donde se manifiesta la necesidad de un usuario de mantener un mayor control sobre toda la información de la cual dispone. El interés en este proyecto de tesis es motivado principalmente por las deficiencias en los buscadores actuales de documentos.

¿Por qué el enfoque semántico?

Las computadoras son máquinas sintácticas y la mera sintaxis no produce semántica. Incluso los mejores buscadores como Google [1] y Yahoo[2] en este momento basan su búsqueda en palabras clave, caso excepcional es el de Hakia[3] y Wolfram Alpha[4], que son buscadores basados en significado; estos últimos trabajan sobre la base del lenguaje natural y sus búsquedas son preguntas concretas. Para responder a ellas se calcula la respuesta más apropiada, pero tales buscadores se enfocan únicamente a la Web.

El uso de semántica en la recuperación y búsqueda de información no es una idea nueva, algunos enfoques consideran, por ejemplo, la representación de un documento mediante ontologías como en los trabajos de Yao *et al.* [69], Ocampo [45] y Shashirekha y Murali [56], entre otros. Estas técnicas han probado que el uso de ontologías es útil para la representación de documentos. Si consideramos el concepto de ontología, que es originalmente una rama de la filosofía, es una conceptualización acerca

[1] Disponible en `http://www.google.com`, accesado en octubre 2011.
[2] Disponible en `http://mx.yahoo.com`, accesado en septiembre 2011.
[3] Disponible en `http://www.hakia.com`, accesado en septiembre del 2011.
[4] Disponible en `http://www.wolframalpha.com`, accesado en septiembre del 2011

de las características básicas de toda la realidad en el mundo [49]. Es evidente que el conjunto de documentos de un usuario común no representa un conocimiento universal, sino mas bien es subjetivo acerca de su realidad y que puede estar muy distante de un solo dominio o de mezclar varios dominios.

Segun Hakia[5] podemos encontrar 10 razones por las que es mejor un buscador semántico y cómo se diferencia de un buscador convencional.

1. Manejo de las variaciones morfológicas. Se espera que un motor de búsqueda semántica maneje todas las variantes morfológicas (como los tiempos verbales, plurales, etc). En otras palabras, los resultados no deben cambiar si se teclea "mejorar, mejora, mejoras, mejorado".

2. Manejo de sinónimos con los sentidos correctos. Con un motor de búsqueda semántica se espera procesar sinónimos (curar, sanar, tratar, etc ..) en el contexto adecuado y con los sentidos de la palabra correcta. Por ejemplo, la palabra "tratamiento" puede significar hacer favores sociales, como en engañar y tratar, que no sería correcto en el sentido médico. El nivel de desambiguación del sentido en un motor de búsqueda semántica es una señal de este progreso.

3. Manejo de las generalizaciones. De un motor de búsqueda semántica se espera que maneje las generalizaciones (enfermedad = ERGE, ELA, SIDA, etc.) donde se expresa la consulta del usuario en términos generales y se espera que el resultado sea específico.

4. Manejo de asociación de conceptos. Tal vez la función más difícil de todas en un motor de búsqueda semántica es reconocer los conceptos y obtener resultados relevantes. Por lo general, la profundidad de esta capacidad es mayor en operaciones verticales y sería poco realista esperar cobertura en todos los temas.

5. Manejo de asociación del conocimiento. Un motor de búsqueda semántica debería incorporar el conocimiento y utilizarlo para obtener resultados relevantes (gripe porcina = H1N1, gripe porcina = influenza.). Lo cual es muy similar al punto anterior ya que coincidir con el

[5]Disponible en `http://company.hakia.com/new/whatis.html`, accesado en septiembre del 2011

conocimiento y combinar conceptos son principios similares, pero diferentes en la práctica en la forma en que la se adquieren.

6. Manejo de consultas y preguntas en lenguaje natural. Se espera que un motor de búsqueda semántica responda con sensatez cuando la consulta se encuentra en una forma de pregunta (qué, dónde, cómo, por qué, etc.). Hay que tener en cuenta que un "motor de búsqueda" es diferente a un sistema de "búsqueda de respuestas". La tarea principal del motor de búsqueda es la de clasificar los resultados de búsqueda de la manera más lógica y relevante, mientras que un sistema de búsqueda de respuestas puede producir un solo fragmento extraído.

7. Capacidad de puntualizar ininterrumpidamente en un párrafo y la frase más relevante. A diferencia de los motores de búsqueda convencionales, en algunos puntos de consulta a los documentos, se espera que la búsqueda semántica sea mucho mejor. Una consulta debe apuntar no sólo a los documentos, sino también a las secciones pertinentes de las mismas. Esto elimina una segunda búsqueda, donde se supone que el usuario abra el documento para encontrar las secciones relevantes.

8. Capacidad de introducir consultas con libertad, sin formatos especiales, como citas u operadores booleanos. Introducir una consulta con requisitos con formato especial se están convirtiendo en una práctica obsoleta, incluso para los actuales motores de búsqueda no semántica. Los formatos especiales realizan grandes aproximaciones para sustituir un significado que concuerde y son signos que revelan la debilidad subyacente de la tecnología de búsqueda.

9. Capacidad para operar sin depender de las estadísticas, el comportamiento del usuario y otros medios artificiales. En un motor de búsqueda semántica se espera obtener resultados relevantes mediante el análisis del contenido de una página (o documento), su origen, los autores y la credibilidad de los resultados en respuesta a una consulta. Basándose en las referencias de enlace, el comportamiento del usuario/etiquetado y otros medios artificiales pueden producir buenos resultados cuando estos datos están disponibles, pero están fuera del ámbito de la búsqueda semántica. Al no depender de la entrada artificial, la tecnología

de búsqueda semántica es más universal, aplicable a cualquier situación, especialmente a los documentos de empresas y contenido en tiempo real en que dichos datos no existen.

10. Capacidad para detectar su propio desempeño. Cuando no hay un análisis de contenido semántico en un algoritmo de búsqueda, los resultados relevantes se refieren a las mediciones de relevancia artificial, al igual que la popularidad de la página. Se espera que un motor de búsqueda semántica produzca un resultado relevante que refleje el grado de correspondencia con el significado. Esta capacidad proporciona flexibilidad para los desarrolladores al aplicar umbrales por significado. En consecuencia, el motor de búsqueda puede entender su pobre desempeño para marcar automáticamente las áreas de mejora que se necesitan.

1.2 Planteamiento del problema

Actualmente los archivos de texto (documentos) constituyen el medio más empleado para plasmar información y conocimiento en todos los ámbitos de la vida moderna. Este hecho representa una gran ventaja sobre el almacenamiento tradicional en papel, sin embargo, las grandes capacidades de almacenamiento de las que se disponen hoy en día afectan negativamente a la eficacia de las búsquedas, ya que los buscadores actuales muestran como resultado grandes cantidades de información de poca calidad. Es decir, devuelven una gran cantidad de documentos o datos que no se relacionan con lo que el usuario esta buscando, de tal forma que se pierde mucho más tiempo en encontrar lo que verdaderamente se necesita. Esto pone en evidencia que las tecnologías de almacenamiento no han crecido a la par con las técnicas de recuperación de información.

Este hecho ha obligado a llevar una organización de los mismos en disco duro o en unidad de almacenamiento *online* para que sea más sencillo buscar y recuperar dicha información. Para ello los usuarios organizan su información personal según un esquema propio y resulta lógico pensar que esta organización implica saber dónde se encuentra cada documento, o al menos tener una idea de cuándo fueron creados o qué información contienen. Sin embargo, estudios realizados por Golemati *et al.* [23]

y Ravasio *et al.* [53] demuestran que los nombres de archivos o las rutas de acceso no aportan ayuda significativa a las búsquedas y las herramientas existentes no facilitan el manejo y recuperación de la información. Incluso tratándose de documentos recientes, las fechas de modificación no resultan ser buenos indicadores para la localización de archivos.

Si consideramos otras herramientas que se han desarrollado como la visualización de archivos con ontologías, experimentos realizados por Katifori [33] utilizando cuatro tipos de ontologías (Class Browser, Jambalaya, TGVizTab y OntoViz), demostraron que los usuarios parecen familiarizarse con el uso de herramientas visuales pero no es el mismo caso cuando se usa una ontología para la búsqueda de documentos.

Otro problema con formas de representación que se han propuesto anteriormente como las ontologías es que no son sencillas y requieren el soporte de especialistas para su desarrollo, además de que se requiere crear diversidad de ontologías para diferentes temas. Un segundo problema que se ha generado con las notaciones semánticas existentes es ¿cómo un usuario podría representar su información en formatos como OWL o RDF?, es decir entre más compleja sea la notación más difícil será para el usuario hacer uso de ellas. Esto exige el desarrollo de herramientas automáticas que faciliten el proceso de transformación de información no estructurada a notación semántica para facilitar a los usuarios la adaptación al nuevo paradigma de la búsqueda semantica.

Además de los problemas mencionados anteriormente tenemos los problemas de descripción del contenido de documentos y representación de consultas formuladas por el usuario. Esta complejidad se asocia al problema del Procesamiento del Lenguaje Natural (NLP), ya que una computadora no es capaz de interpretar expresiones de la forma en como lo hace un ser humano [6].

En el presente proyecto se propone como solución a los problemas planteados anteriormente el uso de herramientas de código abierto con una componente semántica para organizar y buscar grupos de documentos de acuerdo a los temas que se involucren en su contenido. De esta manera cada usuario

podrá contar con una organización personalizada y dinámica de sus documentos.

1.2.1 Hipótesis

Considerando que, usando técnicas de muestreo de texto y relaciones semánticas, el contenido de un documento puede representarse mediante un grafo, es posible mediante un grafo semántico mejorar la organización y búsqueda de documentos.

Esta hipótesis quedará demostrada si la búsqueda retorna mejores resultados y muestra mejores tiempos que las soluciones que existen actualmente. Para comprobar la hipótesis se presentan a continuación los objetivos que se pretenden lograr.

1.3 Objetivos generales y específicos del proyecto

1.3.1 Objetivo General

Obtener un modelo de organización semántica de documentos escritos en inglés mediante grafos.

1.3.2 Objetivos Particulares

Para lograr el objetivo general se proponen los siguientes objetivos específicos.

- Contribuir al estado del arte con una metodología para la representación semántica y búsqueda de documentos en inglés mediante grafos.
- Contribuir al estado del arte con un método de organización semántica de documentos.
- Validar los aspectos de esta propuesta mediante el desarrollo de una aplicación que será que implemente las características del modelo propuesto y que sea probada teniendo en cuenta las métricas de evaluación de la recuperación de información: *precision* y *recall*.

1.4 Contribuciones y resultados esperados

Al finalizar este trabajo se espera:

- La obtención de un modelo de representación de documentos.
- La obtención de un prototipo de software que permita la organización y búsqueda semántica de documentos.
- Una tesis de maestría que describa la representación, implementación y evaluación del modelo obtenido.
- Publicación de la propuesta y resultados obtenidos en algún congreso.

2

Estado del arte

En este capítulo se presenta el marco teórico, el cual permite contextualizar este trabajo de tesis. Para ello se describen algoritmos, métodos, técnicas y herramientas utilizadas para la construcción del grafo semántico.

2.1 Organización de documentos

Según Henderson [28], tanto investigadores como usuarios comunes hoy en día tienen una gran cantidad de documentos digitales por administrar, los cuales en su mayoría emplean algún tipo de software para su organización o plan para ayudarse en esta tarea. Henderson también define la administración de documentos digitales personales como el proceso de adquisición, almacenamiento, gestión, recuperación y uso de documentos digitales. Se menciona que son personales en el sentido de que los documentos son propiedad del usuario y están bajo su control directo, no que necesariamente contienen información sobre el usuario.

Según Blanc–Brude y Scapin [12], el concepto *Personal Information Management* (*PIM* o Gestión de Información Personal) se refiere a un campo de investigación que se centra en el estudio de cómo las personas administran sus documentos físicos (libros, cuadernos, hojas, etc.), así como sus documentos electrónicos (archivos, correos electrónicos, páginas web, etc.) con el objetivo de diseñar herramientas que apoyan la gestión de documentos electrónicos. Las herramientas *PIM* ayudan a que la información que se organiza pueda ser fácilmente recuperada posteriormente. Hoy en día existe una gran variedad de herramientas *PIM* como lo mencionan Bergman *et al.* [8] y Henderson [28], autores que coinciden en que el software actual proporciona un mecanismo para organizar los documentos en un sistema jerárquico de carpetas. Sin embargo, este esquema de organización se adoptó hace más de 40 años para proporcionar un acceso eficaz a los archivos en el disco cuando el usuario tenía pocos documentos. Según Badia [5], la *PIM* no constituyen una idea nueva sino el punto de encuentro en la evolución de varias tendencias: herramientas (gestores de citas, listas de tareas, calendarios), hardware (nuevos aparatos digitales: PDAs, teléfonos inteligentes, tabletas, etc.) y lo más importante, las necesidades. Es decir, un uso intensivo de datos en un mundo rico en información, en el que a veces no encontramos lo que estamos buscando, a pesar de que sabemos que está ahí, en algún lugar.

2.2 Sistemas de Recuperación de Información

Los primeros Sistemas de Recuperación de Información se originaron por la necesidad de organizar la información en repositorios centrales (por ejemplo, bibliotecas). Hoy en día un Sistema de Recuperación de Información, de acuerdo a Kowalski *et al.* [37], consiste en un programa de software que facilita a un usuario encontrar la información que necesita. La información en este contexto puede estar compuesta de texto (incluyendo datos numéricos y de fecha), imágenes, audio, video y otros objetos multimedia. El sistema puede utilizar el hardware estándar o hardware especializado para apoyar a la función de búsqueda y convertir los recursos no textuales en un medio de búsqueda (por ejemplo, la transcripción de audio a texto). El indicador de éxito de un sistema de información

es lo bien que puede minimizar el *overhead* de un usuario para encontrar la información necesaria. *Overhead*, desde la perspectiva de un usuario, es el tiempo requerido para encontrar la información necesaria, excluyendo el tiempo para una lectura de los datos relevantes. Así, la composición de la búsqueda, la ejecución de búsqueda y la lectura de elementos no relevantes son todos aspectos del *overhead* de recuperación de información.

El éxito de un sistema de información es muy subjetivo, con base a qué información es necesaria y la voluntad de un usuario a aceptar el *overhead*. En algunas circunstancias la información necesaria se puede definir como toda la información que está en el sistema y que se relaciona con la necesidad del usuario. En otros casos, se puede definir como la información suficiente en el sistema para completar una tarea, teniendo en cuenta las pérdidas de datos.

Un sistema que admite la recuperación razonable requiere menos funciones que uno que requiera la recuperación completa. En muchos casos la recuperación completa es una característica negativa, porque se sobrecarga al usuario con más información de la que necesita. Esto hace que sea más difícil para el usuario filtrar la información relevante, pero no útil desde los elementos críticos. En la recuperación de la información el término elemento "relevante" se utiliza para representar un elemento que contiene la información necesaria para el usuario.

Desde la perspectiva de un usuario, "relevante" y "necesario" son sinónimos. Desde una perspectiva de sistema, la información podría ser relevante para una declaración de búsqueda (es decir, que cumple los criterios de la sentencia de entrada) a pesar de que no es necesario/relevante para el usuario.

Las dos medidas principales comúnmente asociadas con los sistemas de recuperación de información son *precision* y *recall*, estas métricas se describen en la siguiente sección.

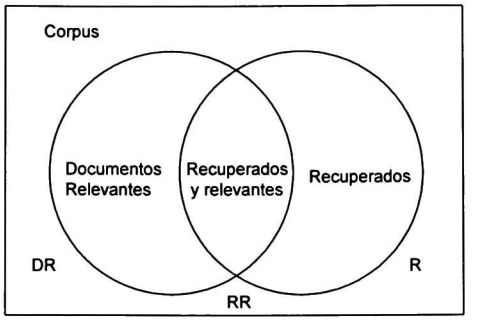


Figura 2.1: Diagrama de Ven que ejemplifica la relación de las métricas *Precision*(RR/R) y *recall*(RR/DR)

2.3 Medidas para la recuperación de información

Un tema importante en el área de Recuperación de Información (IR, por sus siglás en inglés) es la evaluación de los De los sistemas desarrollados para la recuperación de información. La principal razón para evaluarlos es determinar si un sistema realmente ayuda a los usuarios a quienes está destinado. En este sentido, las dos medidas más frecuentes y fundamentales para medir la eficacia de la Recuperación de Información son *precision* y *recall* [29]. En la figura 2.1 se puede observar para una consulta dada por un usuario referente a un tema, cuál sería la proporción de documentos relevantes, documentos recuperados y documentos relevantes que son realmente recuperados.

Así *precision* y *recall* se definen como:

Precision (P): Es la fracción de los documentos recuperados que son relevantes

$$Precision = \frac{\#(Documentos\ Relevantes\ Recuperados)}{\#(Documentos\ Recuperados)} = P(relevantes|recuperados) \quad (2.1)$$

Esta medida responde a la pregunta: ¿Para una búsqueda, qué fracción de los documentos recuperados es relevante?

Recall (R): Es la fracción de documentos relevantes que se recuperan

$$Recall = \frac{\#(Documentos\ Relevantes\ Recuperados)}{\#(Documentos\ Relevantes)} = P(recuperados|relevantes) \quad (2.2)$$

En otras palabras *recall* responde a la pregunta: ¿Para una búsqueda dada, qué fracción de todos los documentos relevantes se han obtenido de todo el conjunto de datos?.

En el contexto de las tareas de clasificación, los términos verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos se utilizan para comparar la calificación de un elemento (la etiqueta de clase asignada al elemento por un clasificador) con la clasificación deseada correcta (la clase o tema al que en realidad pertenece) [43]. Esto se ilustra en el cuadro siguiente:

		resultados correctos/clasificación	
resultado	E1	vp (verdadero positivo)	fp (falso positivo)
obtenido/clasificación	E2	fn (falso negativo)	vn (verdadero negativo)

2.4 Herramientas existentes

A continuación se presentan algunas herramientas relevantes para recuperación de documentos.

2.4.1 Buscadores semánticos

Como se mencionó anteriormente, ya existen herramientas para la búsqueda de documentos electrónicos y en los últimos años ha surgido un interés especial por el enfoque semántico. A continuación se prestan algunas herramientas que se catalogan en esta área.

Zoom. Es un software de indexación y búsqueda en lenguaje natural y análisis documental basado en la comprensión de los contenidos a tratar [1]. La forma en la que trabaja la herramienta Zoom es organizando los documentos alrededor de la información que los estructuran. Para ello cada palabra significativa se inscribe en una cadena de equivalentes semánticos, basta una palabra para encontrar todos los documentos en los que ella o sus equivalentes aparecen. Por ejemplo, una búsqueda a partir de "elecciones" encuentra textos que contienen "candidato", "urna" y "voto", incluso si la palabra "elecciones" no se encuentra explícitamente en el texto. Los equivalentes semánticos constituyen un método que parece funcionar pero el inconveniente que se observa en esta herramienta es que su búsqueda de "equivalentes semánticos" termina por ser una búsqueda por palabras clave, asociadas a la palabra que se busca.

DeepaMehta. Es una plataforma de software inspirada en la idea de un escritorio semántico, es decir, utiliza la tecnología de la Web Semántica en el escritorio para apoyar la Gestión de Información Personal (*PIM*)[2] [55]. Además de proporcionar una interfaz para administrar datos personales también ofrece interfaces para acceder a otras aplicaciones, por ejemplo correos electrónicos, páginas web, notas. El objetivo de DeepaMehta es la gestión del conocimiento, representado a través de una red semántica para manejar colaborativamente el flujo de trabajo y los procesos sociales. La interfaz de usuario DeepaMehta está basada en mapas conceptuales construidos de acuerdo a investigaciones en Psicología Cognitiva. Esta interfaz permite manejar todo tipo de información directa e individualmente, visualizada como un grafo. Su característica principal es la capacidad de manejo de archivos, representando los archivos locales como temas, los temas se unen gráficamente por líneas creando así una relación semántica entre ellos. Una de las desventajas que se observa en esta aplicación es que requiere de la intervención humana para crear los mapas conceptuales.

K-Search. Es una aplicación desarrollada a partir de la propuesta de Bhagdev *et. al.* [10], que consiste en una búsqueda híbrida, una combinación de búsqueda basada en metadatos y palabras

[1] Disponible en `http://www.conocimiento-semantico.com/zoom`, accesado en octubre 2011.
[2] Disponible en `http://www.deepamehta.de/`, accesado en octubre 2011.

clave. Entre otras cosas, K-Search permite:

- La búsqueda de documentos a través de una intranet
- Resumen de la información a través de gráficos
- Exportar la información a hojas de cálculo

Es posible realizar tres tipos de consultas con este enfoque híbrido: *(i)* la búsqueda de semántica pura a través de la identificación única de conceptos/relaciones/instancias, *(ii)* la búsqueda basada en palabras clave en todo el documento y *(iii)* la búsqueda de palabra clave en el contexto. Esta última busca las palabras clave sólo dentro de la porción del documento anotado con un concepto específico o relación. Por ejemplo, dentro del dominio aeroespacial, permite la búsqueda de la cadena "combustible" pero sólo en el contexto de todas las porciones de texto anotadas con el concepto de *"pieza del motor afectada"*. Según sus resultados, el paradigma de búsqueda híbrida se comporta mejor en comparación con las búsquedas basadas sólo en palabras clave. Pero tal como lo indican los mismos autores, estos resultados dependen de la existencia de los metadatos, ya que si un documento no tiene por ejemplo un RDF[3] asociado, la búsqueda será sólo por palabras clave.

SemMF. Es un framework para el cálculo de la similitud semántica entre los objetos que se representan como RDFs u OWL[4]. Toma como entrada un objeto de consulta y una colección de objetos que se compara con el objeto de consulta. Puede ocurrir que no todas las propiedades del objeto sean relevantes, para resolverlo se asignan pesos a las propiedades y las propiedades que se ajustan a la búsqueda se pueden agrupar en temas. Dentro de cada grupo temático se calcula la similitud entre cada propiedad de la consulta y la propiedad de los recursos correspondientes. Estas similitudes se multiplican por los pesos indicados y se suman dando la similitud de *cluster* o grupo. Todas las similitudes del *cluster*, a su vez, multiplicadas por el peso del *cluster* especificado producen la similitud del objeto. La salida es una clasificación de los objetos por los valores de su similitud.

[3]Resource Description Framework

[4]Web Ontology Language. Disponible en `http://semmf.ag-nbi.de/doc/index.html`, accesado en septiembre del 2011.

También proporciona una descripción detallada del proceso de comparación (es decir, los valores de las propiedades del objeto, los valores de similitud para todos los grupos y para cada propiedad del objeto único dentro de un *cluster*, los pesos asociados, etc.) que pueden ser utilizados para generar las explicaciones de la similitud calculada para un objeto. La asignación de pesos representa un problema, puesto que cada propiedad relevante en la consulta RDF se debe especificar explícitamente. Además, si para un valor dado a una propiedad en una consulta se tiene más de una correspondencia (por ejemplo el mismo producto en diferentes colores), se eligirá el que tiene una ponderación mayor.

2.4.2 Analizadores sintácticos para la extracción de texto

Cuando se trabaja con grandes volúmenes de información y variedad de formatos de archivos, el procesamiento de la información se puede convertir en un proceso muy complicado. En estos casos es necesario extraer esa información para que su procesamiento sea más sencillo. Una forma de hacerlo es convertir el contenido de los documentos a una representación en texto plano. Para ello existen algunas herramientas que realizan este trabajo. A continuación se presentan tres ejemplos de herramientas que extraen texto de diferentes tipos de formatos.

Multivalent. Es una aplicación en java y entre sus herramientas se encuentra un extractor de texto[5] Los formatos soportados por esta herramienta incluyen PDF, HTML y DVI entre otros. Esta herramienta es capaz de extraer:

- Texto, opcionalmente normalizado a codificación Unicode
- La estructura de un documento
- Hipervínculos
- Esquemas

[5]Disponible en `http://multivalent.sourceforge.net/`, accesado en octubre del 2011.

- Estilo (tipo de letra, colores)

Text Mining Tool. Es un programa que permite extraer texto de diferentes formatos de archivos como pdf, doc, rtf, chm y html. Está basado en el framework .NET 2.0 y no requiere instalación. Tiene dos modos de operación, interfaz gráfica y por consola, ejecutando *minetext*. A diferencia de las otras herramientas es exclusiva para Windows[6].

Apache Tika. Tika es un programa de Apache Software Foundation y se deriva del proyecto Apache Lucene[7]. En sí, es un conjunto de herramientas para extraer texto y metadatos contenidos en archivos con las siguientes extensiones:

- HyperText Markup Language
- XML y formatos derivados
- Formatos de documentos Microsoft Office
- OpenDocument Format
- Portable Document Format
- Electronic Publication Format
- Rich Text Format
- Compresión y formatos de empaquetado
- Formatos de audio
- Formato de la imagen
- Formatos de video
- Archivos Java class

[6]Disponible en `http://text-mining-tool.com/`, accesado en octubre del 2011.
[7]Disponible en `http://tika.apache.org/`, accesado en octubre del 2011.

- Formato mbox

2.4.3 Etiquetadores

El *part of speech tagging (POS tagging)* también llamado *taggers* o etiquetado de parte de una oración, como lo indican Mohan *et al.* [4], se define como el proceso de asignar a cada palabra en una frase una etiqueta que indica el estado de esa palabra dentro de algún sistema de categorización de las palabras de algún idioma de acuerdo a sus características morfológicas y/o propiedades sintácticas. Los datos marcados se pueden utilizar para la generación de reglas basadas en sistemas de traducción automática que mejoran la precisión. Los taggers se pueden categorizar como basados en reglas o estocásticos. Los etiquetadores basados en reglas usan reglas escritas a mano para distinguir la ambigüedad de etiquetas. Los etiquetadores estocásticos están entre:

- Basados en HMM (*Hidden Markov Model*), es decir, la elección de la secuencia de etiquetas que maximiza el producto de la probabilidad de la palabra.
- Basados en referencia, utilizando árboles de decisión o modelos de máxima entropía para combinar características probabilísticas.

Considerando un enfoque supervisado, POS tagging requiere una gran cantidad de elementos en el corpus[8] etiquetados correctamente.

2.5 Representación de texto

Doan *et al.* [20] consideran que la representación de texto es uno de los pasos más importantes en el procesamiento de texto, como en la recuperación de información, enrutamiento del texto y particularmente en la categorización. También mencionan que de forma general hay dos marcos

[8]Un corpus es una colección de documentos de texto, normalmente en texto plano o en formato SGML (Standard Generalized Markup Language).

comunes para la representación del texto: un modelo de espacio vectorial y un modelo probabilístico. En el modelo de espacio vectorial un documento se considera como un vector en el espacio vectorial en el que cada dimensión corresponde a un término. Aparte del modelo de espacio vectorial, cada documento se considera como un evento en el modelo probabilístico en el que un documento se representa como una bolsa de palabras (*bag-of-words*). En ambos métodos, cada término es ponderado por una función de indexación que caracteriza la importancia de ese término en el documento.

Tal como lo mencionan Lan *et al.* [38], el texto ya está almacenado en forma legible para una máquina, sin embargo formatos como HTML, PDF, DOC, Postscript, etc. generalmente no son siempre convenientes para la mayoría de algoritmos de aprendizaje. Antes de aplicar un método de aprendizaje automático para la categorización de texto, el contenido de un documento se debe convertir a una representación compacta con el fin de ser reconocidos y clasificados por una computadora. La forma más común para representar el contenido de un documento es utilizando sólo un conjunto de términos contenidos en el documento. Entre estos enfoques se encuentra *bag-of-words*, en el que sólo se registra la frecuencia de aparición de una palabra en el documento dejando fuera toda la estructura y orden de las palabras. Sin embargo, aunque pareciera un método relativamente simple presenta el inconveniente de la falta de representación de las relaciones semánticas entre las palabras que se consideran cruciales para el entendimiento humano.

Otra forma de representación abarca múltiples palabras, es decir frases u oraciones. Gracias a esto es posible desarrollar herramientas computacionales lingüísticas para analizar grandes cantidades de texto tomando en cuenta la estructura sintáctica. Por otra parte, para deshacerse del problema de los sinónimos en el lenguaje natural, algunos investigadores han tratado de usar también el significado de las palabras o *clustering* de términos para representar texto. Además de esto, con el fin de captar las relaciones semánticas entre las palabras, algo que se ignora con el uso de *bag-of-words*, algunos autores también incluyen una representación basada en hiperónimos mediante el uso de WordNet[9]

[9]Disponible en `http://wordnet.princeton.edu/`, accesado en septiembre del 2011.

[38].

Otros métodos para la representación de texto se enfocan en la ponderación de términos, en el cual cada término del documento debe estar asociado a un valor o peso. Este valor ayuda a denotar la importancia de este término y a clasificar el documento. En este trabajo se hará uso del algoritmo LDA para representar texto, el cual se describe a continuación.

2.5.1 Latent Dirichlet Allocation

Una descripción del algoritmo Latent Dirichlet Allocation (LDA) la proporcionan Dimai *et al.* [18], en donde se explica que es un modelo probabilístico generativo de un corpus. La idea detrás de esto es que los documentos (documents) se presenten como mezclas al azar sobre temas latentes (*latent topics*). Esta mezcla al azar es utilizada para generar una secuencia de *topics* o temas y palabras (*words*), donde cada tema se caracteriza por una distribución de probabilidad sobre las palabras.

Los elementos básicos del modelo LDA se definen de la siguiente manera:

- Una *word* es un elemento de un conjunto de elementos de V, donde V es la dimensión del vocabulario.
- Un *document* es una secuencia de N palabras denotadas por $\vec{w} = (w_1, w_2, ...w_n, w_N)$, donde w_n es la n-enésima palabra en la secuencia. El orden de las palabras es irrelevante
- Un *corpus* es un conjunto de M documents denotado por $D = \{\vec{w_1}, \vec{w_2}, ...\vec{w_M}\}$
- Un *topic* es un elemento de un conjunto de k elementos

LDA asume el siguiente proceso generativo para un *document* $\vec{w}$, dado k *latent topics* y un vocabulario de palabras V:

1. Elige $N \in Poisson(\xi)$

2. Elige $\theta \in Dir(\alpha)$

3. Para cada una de las N palabras w_n.

   a) Elegir un *topic* $z_n \in$ Multinomial(θ).

   b) Elegir una palabra w_n de $p(w_n|z_n, \beta)$, una probabilidad multinomial condicionada en el *topic* z_n.

Los documentos se generan de forma independiente uno del otro. De las variables involucradas, sólo las palabras w_n se observarán: tópicos z_n y mezclas de tópicos θ que son *latent variables*. La dimensionalidad k y el parámetro α, de la distribución de Dirichlet se supone conocida y fija y la matriz $k \times V\beta$ se supone fija, pero en realidad se desconoce y se debe de estimar. La hipótesis en α puede ser representada como un algoritmo modificado de Newton-Raphson[10] para estimar si la variable α es válida. El número de *topic* k es elegido por el usuario. La i-esima fila de la matriz β es la distribución multinomial de palabras condicionadas en el i-ésimo *topic*. Además, se debe tener en cuenta que la variable auxiliar N es independiente de todos los demás datos de generación variables (θ y $\vec{z}$), por lo tanto, su aleatoriedad podría ser ignorada en ciertas aplicaciones del algoritmo. Algunas otras estrategias de inferencia se pueden adoptar, éstas pueden ir desde los enfoques empíricos hasta un enfoques más completos como el bayesiano (mediante la asignación de las prioridades adecuadas de β y α)

Teniendo en cuenta los parámetros α y β, la distribución conjunta de una mezcla de tópicos θ, un conjunto $\vec{N}$ de tópicos $\vec{z}$ y un conjunto N de palabras $\vec{w}$ viene dada por la Ecuación 2.3.

$$p(\theta, \vec{z}, \vec{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n-1}^{N} p(z_n|\theta)\ p(w_n|z_n, \beta) \tag{2.3}$$

Es decir, la Ecuación 2.3 nos da la probabilidad de un documento dado su contenido haciendo una intersección de las distribuciones de todas las variables conocidas y desconocidas.

[10]El algoritmo modificado de Newton-Rapshson es un algoritmo lineal para el método general de optimización cubica. Este método se utiliza para estimación de máxima verosimilitud de la distribución Dirichlet [44]

Las variables θ = *(mezcla de tópicos)* son muestreadas una por documento y tienen una interpretación semántica (es decir, un vector de proporciones de tópicos en un solo texto, donde los tópicos son descritos por diferentes distribuciones multinominales de palabras). Éstas serán las representaciones numéricas que se utilicen para los documentos.

En la figura 2.2 se puede ver que existen tres niveles de la representación LDA. Los parámetros α y β son parámetros a nivel de corpus, se supone que pueden ser muestreados sólo una vez en el proceso de generación de un corpus. Las variables θ son las variables de nivel de documento, muestreadas una vez por documento. Por último, las variables z_{dn} y w_{dn} son variables a nivel de texto y son muestreadas una vez para cada palabra en cada documento. Las cajas son "placas" que representan repeticiones. La placa exterior representa los documentos, mientras que la placa interior representa la elección repetida de temas y palabras dentro de un documento.

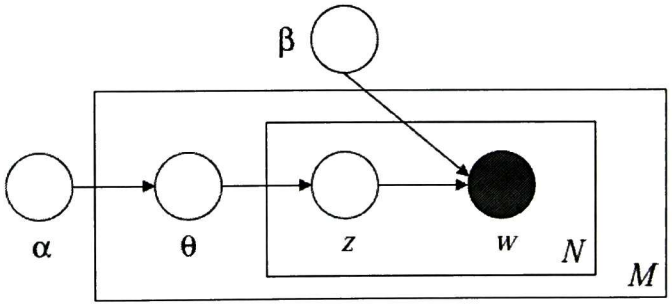


Figura 2.2: Representación gráfica del modelo de LDA.

Es importante distinguir LDA de un modelo de agrupamiento simple que ocurre en una distribución Dirichlet-multinomial. Un modelo de agrupación clásica implicaría un modelo de dos niveles en el cual Dirichlet se muestrea una vez por un corpus, una variable de agrupación multinomial es seleccionada una vez para cada documento en el corpus y un conjunto de palabras se seleccionan para el documento en un agrupamiento variable. Como ocurre con muchos modelos de agrupamiento, cuando un modelo restringe un documento a ser asociado a un solo tema. LDA, por el contrario, consiste en tres niveles y en particular el tema se muestrea en varias ocasiones en el documento. Bajo este modelo, los documentos pueden estar asociados con múltiples temas.

2.6 Agrupamiento de texto

Yan Fu *et al.* [22] mencionan que la minería de texto se ha estudiado en diferentes campos a razón de la gran cantidad y del continuo crecimiento de textos electrónicos, especialmente en la recuperación de la información. Entre las tareas de minería de texto, la agrupación de texto (*clustering* de texto) es una parte muy importante. El objetivo de este tipo de *clustering* es dividir un conjunto de elementos de texto en grupos homogéneos y distinguir precisamente la información verdaderamente relevante. Los métodos estándares para el *clustering* de texto se basan principalmente en el VSM (Vector Space Model).

Además del pre-procesamiento tradicional de documentos, como lo mencionan AlSumait y Domeniconi [2], se puede mejorar la representación de documentos mediante la eliminación de *stopwords* (palabras vacías), quitar palabras raras, *stemming* y normalización, pero aún así sigue siendo esencial integrar la información semántica y los patrones conceptuales a fin de mejorar las capacidades de predicción de los algoritmos de *clustering*. En este sentido existen algunos trabajos relacionados con el *clustering* de texto basado en semántica, tal es el caso del trabajo de Zhonghui *et al.* [72], que proponen un algoritmo de agrupamiento de secuencias semánticas similares, también tenemos el trabajo de Jing *et al.* [32] que usa agrupamiento de texto, medidas de similitud y frecuencias para relacionar las palabras en el texto semánticamente. Otros enfoques realizan agrupación de texto usando frases completas, como el trabajo desarrollado por Shehata [57]

En la subsección siguiente se describe el algoritmo de *clustering* de textos CBC, que es un algoritmo de *clustering* que se emplea y aplica en este proyecto para agrupar términos y la red SOM para reducir la dimensionalidad de los datos.

2.6.1 Clustering By Committee

Clustering By Committee (CBC - Agrupamiento por Comités) es un algoritmo de agrupamiento que tiene como objetivo producir grupos de calidad en *clustering* de documentos. CBC construye el centroide de un grupo considerando el promedio de los vectores de características de un subconjunto de los miembros del *cluster*. El subconjunto es visto como un comité que determina qué otros elementos pertenecen a la agrupación. Eligiendo cuidadosamente los miembros del comité, las características del centroide tienden a ser las características más típicas de la clase.

CBC representa elementos como vectores de características. Las características de un documento son los términos (por lo general palabras sueltas) que se producen dentro de él y el valor de una función que es una estadística del término. Por ejemplo, la estadística puede ser simplemente la frecuencia del término, *tf (term frequency)*, en el documento. Con el fin de reducir términos con un bajo poder discriminante *tf* normalmente se combina con la inversa de la frecuencia del término en el documento, la *idf (inverse document frequency)*, que es el inverso del porcentaje de documentos en los que el término se produce. Esta medida se conoce como *tf-idf* y se representan mediante la Ecuación 2.4.

$$tf - idf(t,d) = tf(t,d) \times idf(t) \tag{2.4}$$

donde $tf(t,f)$ representa la frecuencia con que el término t aparece en el documento d. $idf(t)$ es la frecuencia inversa de los documentos en donde aparece el término t.

En el algoritmo CBC, para cada elemento e se construye un vector de conteo de frecuencia $C(e) = (c_{e1}, c_{e2}, ..., c_{em})$, donde m es el número total de funciones y c_{ef} es el conteo de la frecuencia de la función f que ocurre en el elemento e. En la agrupación de documentos, e es un documento y c_{ef} es la frecuencia de los términos de f en e. Con ello se construye una información vectorial mutua $MI(E) = (mi_{e1}, mi_{e2}, ..., mi_{em})$, donde mi_{ef} es la información mutua entre los elementos e

y características f, que se define como:

$$mi_{ef} = \log \frac{\frac{c_{ef}}{N}}{\frac{\sum_i c_{if}}{N} \times \frac{\sum_j c_{ej}}{N}} \tag{2.5}$$

donde $N = \sum_i \sum_j c_{ij}$ es el conteo de la frecuencia total de todas las características de todos los elementos.

Se calcula la similitud entre dos elementos ei y ej utilizando el coeficiente del coseno de sus vectores de información mutua:

$$sim\,(e_i, e_j) = \frac{\sum_i mi_{e_i f} \times mi_{e_i f}}{\sqrt{\sum_i mi^2_{e_i f} \times mi^2_{e_i f}}} \tag{2.6}$$

Este algoritmo se compone de tres fases, las cuales se describen a continuación:

Fase I: Buscar los elementos superiores similares

Se calcula la matriz de similitud completa entre los pares de elementos, la cual es cuadrática. Sin embargo, es posible reducir dramáticamente el tiempo de ejecución aprovechando el hecho de que el vector de características es escaso. Por las características de indexación se puede recuperar el conjunto de elementos que tienen una determinada característica. Para calcular los elementos de similaridad de un elemento e, primero es necesario ordenar la información vectorial mutua $MI(e)$ y considerar sólo un subconjunto de las características de mayor información mutua. Por último se debe calcular la similitud entre pares e y los elementos que comparten una característica de este subgrupo.

Fase II: Buscar comités

Recursivamente se busca ajustar los grupos dispersos. En cada paso recursivo el algoritmo encuentra un conjunto de *clusters* estrecho, llamados comités e identifica los elementos que quedan como residuos ya que no están cubiertos por ningún comité. Se dice que un comité *cubre* un elemento si los elementos similares al centroide de un comité exceden un cierto umbral de similitud. Mediante el

algoritmo se intenta entonces recursivamente encontrar más comités entre los elementos que quedan como residuos. La salida del algoritmo es la unión de todos los comités que se encuentran en cada paso recursivo. Los detalles de la fase II se presentan en el algoritmo 1. En el paso 1 el puntaje refleja una preferencia por las agrupaciones más grandes y más fuertes. El paso 2 da preferencia a los grupos de mayor calidad. En el paso 3, un *cluster* sólo se mantiene si éste es similar a algún grupo, además de ello debe estar por debajo de un umbral fijo. En el paso 4 termina la recursividad si no se encuentra un comité en el paso anterior. Los residuos de elementos se identifican en el paso 5 y si no se encuentran los residuos el algoritmo termina, de lo contrario, se aplicará el algoritmo de forma recursiva a los elementos que quedan como residuos. Cada comité que se descubre en esta fase define uno de los grupos de salida final del algoritmo.

Fase III: Asignar elementos a las agrupaciones

Cada elemento es asignado al grupo que contiene el comité al que es más similar. Esta fase se parece al algoritmo *K*-means ya que se asigna cada elemento a su centro de gravedad más cercano. En esta fase, a diferencia de *K*-means, el número de *clusters* no es fijo y los centroides no cambian (es decir, cuando un elemento se agrega a un *cluster*, no se añade al comité del *cluster*).

El algoritmo CBC se ha utilizado para la clasificación de texto, [71] y [48] son ejemplos de uso de este algoritmo.

2.6.2 Self-Organizing Map

Las redes SOM (Self-Organizing Map), como las describen Bodyanskiy *et al.* [11], es un tipo de red neuronal que fue propuesta por Kohonen [36] en 1982. Se utilizan ampliamente en problemas de minería de datos y procesamiento inteligente de datos, tales como la agrupación, diagnóstico, compresión de información, etc. Ejemplos de estos usos se pueden observar en los trabajos de [68], [50] y [3] que utilizan una red SOM para la clasificación de texto.

Algoritmo 1 Fase II de CBC

Entrada: Una lista de elementos de E que se agrupan, una base de datos de similaridad S de la fase I, los umbrales de θ_1 y θ_2.

Paso 1: Para cada elemento $e \in E$
Agrupar los elementos superiores similares de e para S utilizando el de enlace promedio del *clustering*.

Para cada *cluster* descubierto c calcular la puntuación siguiente: $|c| \times avgsim(c)$, donde $|c|$ es el número de elementos en c y $avgsim(c)$ es la similitud promedio de pares entre los elementos de c.
Almacenar el *cluster* de mayor puntuación en una lista L.

Paso 2: Clasificar los *clusters* de L en orden decreciente de sus puntajes.

Paso 3: Sea C una lista de los comités, inicialmente vacía.
Para cada *cluster* $c \in L$ clasificados en orden:
Calcular el centroide de c para el promedio de la frecuencia de los vectores de sus elementos y calcular el vector de información mutua del centroide de la misma manera como se realizó para los elementos individuales.
Si c es similar al centroide de cada comité agregado previamente a C y está por debajo de un umbral θ_1, agregar c a C.

Paso 4: Si C está vacío, terminar y retornar C.

Paso 5: Para cada elemento $e \in E$:
Si e es similar a todos los comités en C y es inferior al umbral de θ_2, añadir e a la lista de residuos R.

Paso 6: Si R está vacío, terminar y retornar C
De lo contrario, retornar la unión de C y la salida de una llamada recursiva a la Fase II utilizando la misma entrada, excepto sustituir E con R.

Salida: una lista de comités

Desde el punto de vista de Wünstel *et al.* [66], SOM es un método de análisis de datos inspirado en la estructura de ciertos tipos de corteza en el cerebro. Es capaz de identificar diferentes *clusters* de datos de gran dimensión y proyectar ("trazar un mapa") los datos de los diferentes *clusters* a una cuadrícula de dos dimensiones respetando la topología, es decir, la estructura de vecindad de los

datos. Así se puede obtener fácilmente una visualización de los datos. El mapeo y la capacidad de visualización es una ventaja de la red SOM, en comparación con métodos estadísticos estándares. En particular SOM no sólo separa diferentes *clusters*, sino que también resuelve la estructura interna de los *clusters*. Matemáticamente separar la estructura interna de los *clusters* se relaciona con los modelos de la superficie principal para distribuciones de datos conocidos en estadística.

Para entrenar una SOM cada punto de datos del espacio de grandes dimensiones se proyecta sobre un elemento de la red SOM de dos dimensiones, es decir una unidad de SOM. Los puntos de datos cercanos se proyectan a la misma unidad o una unidad cerca en la red. A su vez, cada unidad SOM representa un vector en el espacio de datos de alta dimensión, de tal manera que la SOM puede ser considerada como una inmersión de la red SOM de dos dimensiones en el espacio de datos de alta dimensión.

2.7 Cálculo de la similitud semántica

Existe una serie de enfoques para el cálculo de la similitud semántica que podrían ser útiles en una herramienta de predicción. Estos sistemas suelen ser discutidos en el contexto de la Lingüística Computacional. Los enfoques, en términos generales, se pueden clasificar de la siguiente manera [34]:

Estadísticos

Las relaciones semánticas entre los términos son capturados para la probabilidad de su co-ocurrencia en un corpus textual. Algunos de estos sistemas estadísticos se basan en un espacio de representación vectorial de una lengua. Algunos enfoques estadísticos relevantes son Latent Semantic Analysis (LSA) e Hyperspace Analogue to Language (HAL). En HAL y LSA los términos están representados por vectores en un espacio semántico. Los vectores límite que están más cerca en un espacio semántico multidimensional se consideran que tienen una mayor relación semántica. La relación semántica puede incluir similitud (sinonimia total o parcial), meronimia (entre una parte y

el todo), hiperonimia o relaciones ES-UN (perro ES-UN animal) y las relaciones causa-efecto.

Los métodos puramente semánticos tienen algunas limitaciones potenciales entre las que se encuentran:

- El tipo de relación semántica no es detectado.
- Diferentes "sentidos" de las palabras son tratadas como una sola (por ejemplo, un banco puede referirse a una entidad financiera o un mueble). Las diferentes formas de una palabra se tratan como palabras separadas (tales como la condición, condicional y acondicionamiento).
- Parte de la oración no se tiene en cuenta

Taxonómicos

El contenido y las relaciones de una jerarquía de términos se utilizan para derivar un valor cuantitativo de similitud entre los términos. Los términos están organizados por la subsunción de los conceptos más generales subsumiendo los más específicos (por ejemplo, el automóvil subsume coche). Otras relaciones también se definen (por ejemplo, el volante es una parte de un coche).

Las relaciones taxonómicas se utilizan para derivar un número de similitud semántica y las medidas de la relación. Tal vez el concepto de taxonomía más conocido es WordNet, que es esencialmente una base de datos léxica que codifica las relaciones de sinonimia, hiperonimia y meronimia de las palabras del idioma inglés. CYC[11] es también una gran base de conocimiento y motor de razonamiento de sentido común con una taxonomía de conceptos y relaciones tales como la causalidad.

Híbridos

Este enfoque toma un poco de los dos enfoques anteriores. Se basa en una representación taxonómica de mejora de conceptos de las propiedades estadísticas de un corpus de texto.

[11] Disponible en `http://www.opencyc.org/`, accesado en octubre del 2011.

2.8 Hipótesis Distribucional

Una interesante alternativa es derivar el conocimiento de textos mediante el análisis de cómo se usan ciertos términos en lugar de buscar su definición explícita. En este sentido la hipótesis distribucional [25] asume que los términos son similares en la medida en que comparten similares contextos lingüísticos [16].

La hipótesis distribucional se fundamenta en que cualquier lenguaje tiene restricciones para que las palabras aparezcan juntas en una misma sentencia. Para los sustantivos hay un conjunto restringido de verbos con los cuales ocurren, por lo que cada sustantivo puede ser caracterizado de acuerdo a los verbos con los que ocurre, este hecho permite agruparlos o clasificarlos.

La hipótesis distribucional modela la similaridad semántica usando información contextual. Dado un conjunto de contextos V extraídos de un corpus C, se define una función que asocia los contextos a cada subconjunto de palabras (Ecuación 2.7)

$$C : 2^W \rightarrow 2^V \tag{2.7}$$

El contexto puede estar formado por los verbos o adjetivos que ocurren (están presentes) en el corpus, a los cuales nos referiremos en adelante como *contexto*. La similaridad sim_w entre dos conjuntos de palabras W_1 y W_2 se obtiene como la similaridad sim_V entre conjuntos relacionados mediante sus contextos (Ecuación 2.8):

$$sim_w(W_1, W_2) \approx sim_V(C(W_1), C(W_2)) \tag{2.8}$$

En este trabajo se determina la similaridad extrayendo del corpus relaciones verbo/sujeto, verbo/objeto y verbo/complemento contenidas en las sentencias de los documentos.

2.9 WordNet

WordNet es una gran base de datos léxica del idioma inglés, aunque se ha extendido a otros lenguajes mediante el proyecto EuroWordNet[12]. Esta base de datos contiene un conjunto de sinónimos cognitivos llamados *synsets*. Los *synsets* son grupos de sustantivos, verbos, adjetivos y adverbios cuyo objetivo es expresar significados. Además de los significados que determinan los *synsets*, éstos se vinculan mediante relaciones conceptuales semánticas y léxicas, dando como resultado una red a través de la cual es posible navegar entre los significados.

El desarrollo de la base de datos léxica de WordNet requiere realizar dos tareas principales para la construcción de la misma, es así como lo describe Beckwith *et al.* [7]. Una de ellas consiste en escribir los archivos de recursos que contienen la base de datos léxica, que es la esencia de WordNet. La segunda tarea es la creación de un conjunto de programas para computadoras que acepten los archivos de recursos. En el desarrollo de estas tareas el sistema completo de WordNet se divide en cuatro partes: los archivos lexicógrafos fuente de WordNet; el software para convertir estos archivos en la base de datos léxica WordNet; la base de datos léxica WordNet; la suite de herramientas de software utilizado para acceder a la base de datos.

En el desarrollo de esta tesis se utilizaron algunas de las herramientas para el acceso a la base de datos WordNet. Estas herramientas son un conjunto de proyectos adicionales entre los que se encuentran:

- Redes Semánticas – WordNet en idiomas distintos del Inglés
- Interfaces Web – Proporcionan acceso a WordNet a través de la red
- Interfaces locales - (APIs) – Un conjunto de archivos disponibles para su descarga y codificados en diferentes lenguajes de programación.

[12]Disponible en `http://www.illc.uva.nl/EuroWordNet/\#EuroWordnet`, accesado en septiembre de 2011.

- Extensiones – Son sistemas que amplían las características de WordNet o la integran en sistemas más grandes.
- Asignaciones – claves para sentidos y/o mapeo de synset entre diferentes versiones de WordNet

Específicamente se utilizaron las APIs JWNL[13] (Java WordNet Library) y RiTa[14], que son APIs de Java para acceder a WordNet y proporcionan acceso a nivel de API para los datos de WordNet. Están escritas en Java puro (no utiliza código nativo), por lo que son totalmente portátiles.

WordNet se ha utilizado en trabajos como el desarrollo de ontologías, algunos ejemplos de esto son los trabajos de Ocampo [45] y Hu *et al.* [30]. También se ha utilizado en la categorización de texto como en el trabajo de Li *et al.* [40], para el *clustering* de documentos en trabajos como el de Chen *et al.* [15] y propuestas para la clasificación de texto como la de Liu *et al.* [41].

2.10 Trabajos relacionados

En esta sección se presentan los trabajos relacionados que se considera más relevantes para este trabajo de tesis, en lo que se refiere a la representación de documentos utilizando grafos.

El enfoque de Shi *et al.* [58] es un método de clasificación semántica. Éste representa los documentos semánticos conectando los conceptos como un grafo y calculado la similaridad del documento para editar la distancia entre las conexiones del grafo de conceptos. Utiliza únicamente dos relaciones semánticas: hiperónimo y la relación *es-parte-de*. A diferencia de lo que se propone para esta tesis en comparación con el trabajo de Shi *et al.*, que prueba su enfoque a partir de documentos semánticos como RDFs y ontologías, en esta propuesta de tesis se crea el documento semántico

[13]Disponible en `http://sourceforge.net/projects/jwordnet/`, accesado en septiembre de 2011

[14]Disponible en `http://rednoise.org/rita/wordnet/documentation/index.htm`, accesado en septiembre de 2011

desde un documento común, sin ningún tipo de información semántica. Además de esta diferencia hay que mencionar que aunque este trabajo de tesis y Shi *et al.* [58] proponen una representación basada en grafos para un documento, Shi *et al.* sólo consideran dos tipos de relaciones, en cambio en esta propuesta de tesis se consideraran relaciones del tipo: sinónimos, hiperónimos e hipónimos, esto con el fin de abarcar no sólo las relaciones del propio documento, sino también las relaciones que se pueden tener con conceptos que se encuentran en diferentes documentos.

Existen otras propuestas basadas en grafos como la de Wang y Taylor [61], ellos manejan el término "bosque de conceptos" que se construye con ayuda de una ontología y la base de datos léxica de WordNet. El primer paso en este enfoque es obtener la frecuencia de ocurrencia de las palabras en el documento y la eliminación de las *stopwords*. Hay que destacar que este trabajo sólo considera la relación de hiperónimo entre las palabras del documento. El bosque de conceptos se construye tomando dos términos del mismo documento (T1 y T2), si los *synsets* que corresponden a dichos términos, llamados S1 y S2, tienen una relación de hiperonimia, entonces se utilizará el synsetIDs para representar los conceptos de T1 y T2 respectivamente y los demás sentidos para T1 y T2 serán descartados. Este proceso se completa cuando todos los pares de palabras se investigan. Siguiendo los pasos anteriores se puede llegar a construir un árbol de conceptos que pueden contener términos o synsetIDs que no están estrechamente relacionados con los temas principales del documento. Para solucionar este problema se utilizan las frecuencias de los términos que ocurren en el documento para calcular un índice de contenido semántico para un árbol de conceptos en el bosque de conceptos, es decir, el valor de frecuencia de las palabras, para eliminar los árboles con frecuencias más bajas. Por último se mide la similitud semántica de dos documentos de texto sólo comparando los conjuntos de términos de sus respectivos bosques de conceptos. Una desventaja de este enfoque es que sólo parece comportarse mejor que los métodos VSM (Vector Space Model) y LSI (Latent Semantic Indexing) cuando se trata con documentos pequeños.

Otro trabajo interesante es el de Malo *et al.* [42] que se resume como un modelo basado en reglas de filtrado de criterios, donde un documento recuperado puede contener conceptos que

están relacionadas con los conceptos de la consulta siempre y cuando exista una regla previamente establecida que las relaciones. Los autores tratan de relacionar el tema de un documento con otros problemas asociados con el contenido de Wikipedia y no el contenido real del documento a diferencia de lo que trata el enfoque de esta tesis. Tampoco es totalmente autónomo en contraste con nuestro enfoque, ya que el usuario debe proporcionar una descripción del tema, los conceptos centrales y un pequeño conjunto de documentos de ejemplo.

2.11 Resumen

En este capítulo se han descrito de manera general los sistemas de recuperación de información, que es donde se contextualiza este trabajo de tesis. También se presentaron algunas herramientas para la recuperación de información con los que se tiene cierta relación. Asimismo se describieron los conceptos básicos para la representación de texto.

Se incluyó también una descripción de los algoritmos de agrupamiento de texto que se exploraron para la realización de este trabajo. Asimismo, se describió la base para la representación de texto empleada y la base de datos léxica WordNet.

Para finalizar se presentaron algunos trabajos relacionados con el trabajo desarrollado en esta tesis. Particularmente se describieron tres trabajos encontrados en la literatura, que tienen una mayor relación con el trabajo que se presenta.

3

Metodología

En este capítulo se presenta la metodología propuesta para la construcción de un grafo semántico a partir de un conjunto de documentos escritos en inglés y una fórmula para el cálculo de similitud sobre dicho grafo.

Como hemos visto en el capítulo anterior, ya existen muchas propuestas relacionadas con la búsqueda semántica, pero el problema no se basa sólo en desarrollar un método de búsqueda, consideramos necesario proponer también una nueva notación o estructura semántica que cubra las deficiencias que presentan la propuestas existentes.

La idea del presente trabajo es definir un proceso de indexación automática que refleje relaciones semánticas construidas a partir de un conjunto de archivos, además de un método de búsqueda que no sólo considere los criterios introducidos por un usuario sino que también tome en cuenta la información semántica que se puede extraer de una representación con grafos. Para cumplir estos objetivos se diseñó la metodología que se presenta en las secciones 3.1 a 3.3, en las cuales se describe el proceso de indexación, representación semántica y búsqueda.

3.1 Metodología para la construcción del grafo semántico

La mayoría de las aplicaciones enfocadas a la indexación de datos o búsqueda optan por crear aplicaciones semiautomáticas las cuales requieren la definición de parámetros para su funcionamiento o en su defecto fijan parámetros estáticos. El objetivo de usar diversos algoritmos utilizados en el área de minería de datos en conjunto, es que se llegue a una representación semántica dinámica en la que cada metagrafo se derive de las características del conjunto de datos de entrada, dado que cada grupo de datos puede ser diferente. También se busca que todo el proceso de notación semántica se vuelva transparente para el usuario, como comentábamos en el planteamiento del problema, el entendimiento de notaciones complejas, algoritmos y la intervención obligada del usuario para el uso de una metodología es una de las razones que impide mayoritariamente la adopción del nuevo paradigma de la búsqueda semántica.

En la Figura 3.1 tenemos las faces del preprocesamiento del texto y aplicación de diversos algoritmos que se requieren para extraer ciertas características del texto, cabe hacer notar que cada fase extrae características diferentes para el grupo de datos de entrada. La salida de cada uno de estos algoritmos es la estrada para la siguiente fase, ya que si se utilizara sólo uno o algunos de estos algoritmo no podríamos obtener todos los elementos necesarios para llegar a la representación que se desea.

Con esta metodología también se deja abierta la posibilidad a otros investigadores para hacer uso de ella para otros fines. Por ejemplo existen aplicaciones basadas en LDA, cuyo principal inconveniente es la sintonización de parámetros de LDA. En esta metodología se sugiere el uso del algoritmo SOM para obtener esos parámetros sin intervención del usuario, y se explica detalladamente la configuración de ambos algoritmos para que este proceso sea recreado sin necesidad de algún programa o *framework* en específico.

Para sintetizar, la principal aportación de la metodología propuesta en la Figura 3.1 es utilizar

las características definidas por el conjunto de algoritmos para crear enlaces entre los nodos de un grafo para que la información sea más representativa semánticamente pero menos compleja que las propuestas actuales, como las ontologías.

Recordemos que la indexación o representación de los datos es sólo uno de los problemas que implica la Recuperación de Información, por ello a lo largo de ese capítulo se describirán detalladamente no sólo las fases para la representación si no también los métodos de búsqueda ideados para esta propuesta.

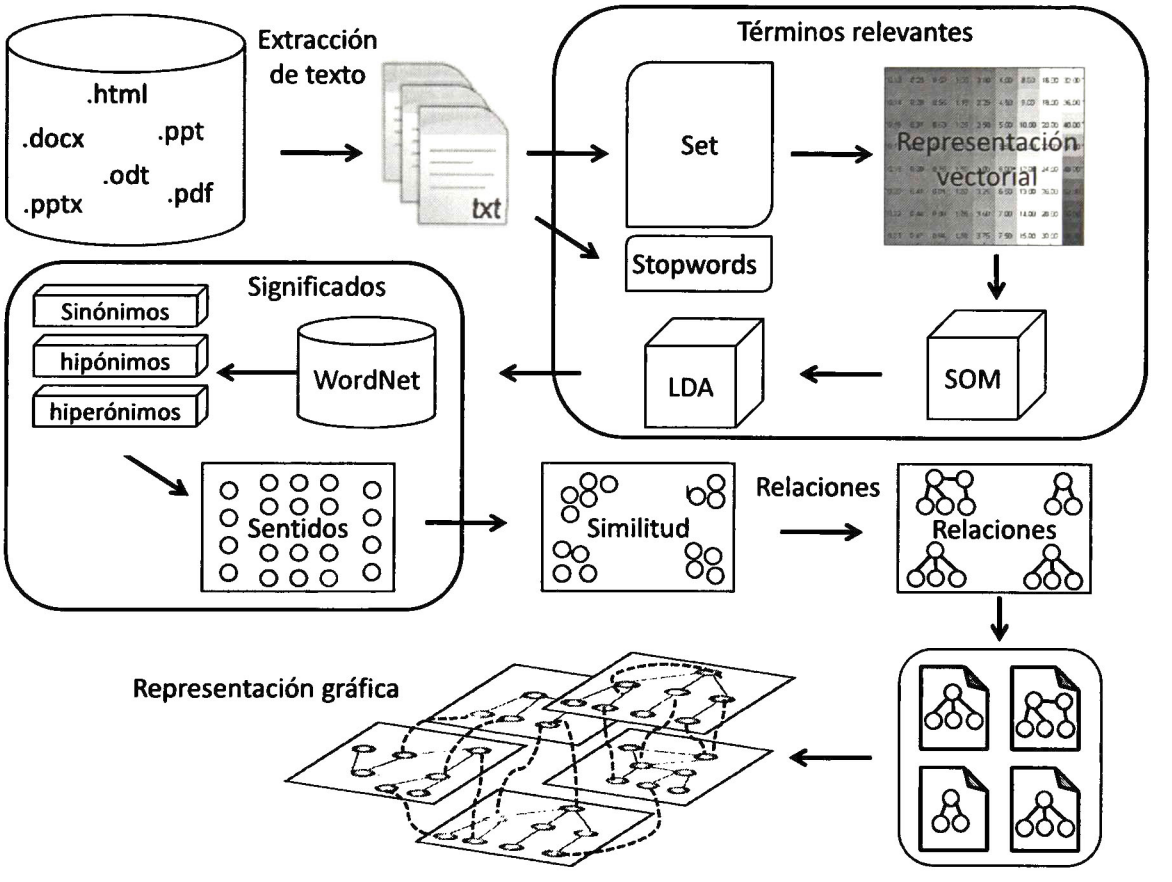


Figura 3.1: Metodología propuesta para la construcción de un grafo semántico

- **Extracción de texto.** Se realiza la lectura de los documentos y su transcripción a texto plano.

- **Extracción de términos relevantes.** Esta etapa comprende la extracción de los términos más significativos de todos los documentos basado en la hipótesis distribucional.

- **Recuperación de significados**. Se debe encontrar el significado de las palabras contenidas en los documentos.

- **Obtención de relaciones**. Permite encontrar las relaciones que pueden existir entre diferentes palabras y agrupar las que sean similares.

- **Representación gráfica**. Se crea un grafo que represente las relaciones entre las palabras contenidas en los diferentes documentos.

A continuación se desglosan las actividades de las etapas arriba mencionadas.

3.1.1 Etapa I: Extracción de texto

El primer paso es transformar el contenido de los documentos que se encuentren en formatos como "docx", "xlsx", "pptx", "doc", "xls", "ppt", "pdf", "odt", "odp", "ods", "odg", "ps", "ppt", "html", "xml", "zip", "tgz", "tar", "gz", "rtf" y "pas" a su representación en texto plano. Esto permite trabajar con diferentes tipos de archivos de origen.

Esta transformación es necesaria ya que es más fácil manipular texto plano que texto en diferentes formatos. Adicionalmente este paso también se requiere para no mover ni alterar físicamente los documentos de entrada. Para esto se pueden usar herramientas parser (en el caso de esta tesis se utilizó Apache Tika).

3.1.2 Etapa II: Extracción de términos relevantes

3.1.2.1. Extracción de términos

El siguiente paso es eliminar los *stopwords*. Estas palabras no son consideradas como relevantes para el contenido del archivo y tampoco ayudan a la búsqueda, por el contrario se eliminan o filtran en la mayoría de los sistemas de recuperación de información.

En este paso también se hace una extracción de las relaciones verbo-sustantivo, descartando el resto de las palabras. Para ello se emplea la Semántica Distribucional, que es la rama de Procesamiento de Lenguaje Natural que se definió como un intento por modelar el significado de las palabras, frases y documentos en base a la distribución y uso de las palabras en un corpus de texto, con el fin de aproximarse mas al significado real de los términos [64]. En este sentido la hipótesis distribucional vista en la sección 2.8 nos dice que los verbos y sustantivos aportan mayor significado a una oración. Por ejemplo, teniendo en cuenta los siguientes contextos:

1. *The X sleeps near the wooden fence.*

2. *The X chews the grass in the meadow.*

3. *The farmer shears the white X.*

El conjunto de palabras {*sleep, fence, chew, grass, meadow, farmer, shear, white*} que constituye una representación simplificada de los contextos acumulados de *X*, proporciona información suficiente (al menos para un ser humano) para identificar a *X* como una *"sheep"* [9]. Es por ello que se han elegido a los verbos y sustantivos como términos mas relevantes, descartando así el resto de los elementos de una oración. Para esto se obtienen las dependencias existentes de cada oración mediante el uso de un analizador gramático-probabilístico y un etiquetador cuya función se describe en la sección 2.4.3.

Con el uso del analizador gramatical obtenemos un mapa de dependencias que provee una descripción simple y uniforme de las relaciones existentes en una oración. De acuerdo a Marneffe y Manning [17], dicha descripción debe permitir a un ser humano, sin conocimientos lingüísticos, interpretar las relaciones de manera fácil y efectiva. Teniendo el mapa de dependencias se construyen todos los posibles árboles de la oración de entrada y el analizador devuelve el árbol con más alta probabilidad.

Un analizador que cumple con las características anteriores es el *Stanford Parser*[1], que considera 52 relaciones gramaticales. Sin embargo, se eligió sólo un pequeño conjunto de dependencias para la extracción de relaciones ya que el enfoque propuesto está basado en la hipótesis distribucional explicada en la sección 2.8. El tipo de dependencias son relaciones binarias, donde una relación gramatical tiene un *gobernador* (también conocido como regente o cabeza) y un dependiente. Las dependencias utilizadas han sido tomadas de Marneffe y Manning [17] y son las siguientes:

csubj: cláusula del sujeto

Es el sujeto sintáctico clausal de una cláusula, es decir, el sujeto es en sí mismo una cláusula. El *gobernador* de esta relación no siempre será un verbo. Ejemplo:

What she said makes sense $\Rightarrow$ *csubj(makes, said)*

csubjpass: sujeto clausal pasivo

Un sujeto clausal pasivo es un sujeto sintáctico clausal de una cláusula pasiva. Ejemplo:

That she lied was suspected by everyone $\Rightarrow$ *csubjpass(suspected, lied)*

[1]Disponible en `http://nlp.stanford.edu/software/lex-parser.shtml\#Download`, accesado en septiembre del 2011

dobj: objeto directo

El objeto directo de una VP (verbal phase-frase verbal) es la oración nominal que es el objeto del verbo; el objeto directo de una cláusula es el objeto directo del VP, el cual es el predicado de la cláusula. Ejemplo:

She gave me a raise ⇒ *dobj(gave, raise)*

iobj: objeto indirecto

El objeto indirecto de una VP es la oración nominal que es el objeto del verbo; el objeto indirecto de una cláusula es el objeto indirecto del VP, el cual es el predicado de la cláusula. Ejemplo:

She gave me a raise ⇒ *iobj(gave, me)*

subj: sujeto nominal

Un sujeto nominal es una oración nominal que es el sujeto sintáctico de una cláusula. El *gobernador* de esta relación podría no ser un verbo. Ejemplo:

Clinton defeated Dole ⇒ *nsubj (defeated, Clinton)*

nsubjpass: sujeto nominal pasivo

Un sujeto nominal pasivo es una oración nominal que es el sujeto sintáctico de una clausula pasiva. El *gobernador* de esta relación podría no ser un verbo. Ejemplo:

Dole was defeated by Clinton ⇒ *nsubjpass(defeated, Dole)*

xsubj: controlando el sujeto

Un sujeto de control es la relación entre la cabeza de una cláusula complemento y el sujeto externo de la cláusula. Ejemplo:

Tom likes to eat fish ⇒ *xsubj (eat, Tom)*

El etiquetador empleado indica el sentido gramatical de cada palabra contenida en la oración, es decir etiqueta cada palabra como un sustantivo o verbo.

Del total de relaciones identificadas en el corpus se obtiene un subconjunto R que satisface dos restricciones [45]:

1. Las relaciones coinciden con al menos 7 de los tipos de dependencias obtenidas por el analizador.
2. Las relaciones contienen la palabra *gobernadora* etiquetada como verbo y la palabra dependiente etiquetada como sustantivo, obtenidas por el etiquetador.

A cada elemento del conjunto R se aplica el proceso de lematización (o stemming)[2]. El objetivo de aplicar este proceso es aumentar la frecuencia de ocurrencia de un mismo término. Así encontraremos con qué ocurrencia se encuentra a un verbo con un determinado sustantivo.

3.1.2.2. Representación vectorial

Con el proceso anterior se obtiene una representación vectorial con base al peso de una relación verbo-sustantivo, calculando para ello el valor de *tf-idf*. El valor *tf* es la *frecuencia del término* en el segmento de texto. En este enfoque para el cálculo de *tf-idf*, un término se refiere a la relación verbo-sustantivo, es decir se cuenta el número de veces que aparece un sustantivo acompañado de un verbo determinado. El valor de *idf* se utiliza para indicar la distinción del término. Esto se traduce en mayor peso para términos que aparecen con más frecuencia y mayor peso para los términos inusuales [67]. La idea de una expresión *tf-idf* es que el peso de las relaciones deba reflejar la importancia relativa de una relación en un documento con respecto a las otras relaciones en el documento.

La ponderación *tf-idf* es un valor usado a menudo en tareas de recuperación de información y minería de texto. Este peso es una medida estadística usada para evaluar cuan importante es una palabra respecto a un documento perteneciente a una colección de documentos, y evitar así el uso de valores absolutos en términos con mucha frecuencia pero que no aportan realmente un significado

[2]**lematización** es un proceso mediante el cual la terminacion de las palabras u otros afijos son eliminados o modificados a fin de que las palabras que difieran en formas no relevantes pudiendo fusionarse y tratarse como equivalentes. Un programa de computadora que lleva a cabo esa transformación se conoce como un *stemmer* o *stemming algorithm*. La salida de un *stemming algorithm* se conoce como un lema [46].

importante.

La fórmula para calcular la frecuencia de los términos se muestra en la Ecuación 3.1 [73]:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3.1}$$

donde $n_{i,j}$ es el número de veces que aparece el término t_i en el documento d_j, y el denominador es la suma del número de ocurrencias de todos los términos en el documento d_j. Por desgracia, esta frecuencia no es una probabilidad realista, por esto se utiliza para incrementar la precisión otra estimación, *la frecuencia de los términos normalizada*. El valor *tf-idf* es usualmente normalizado para prevenir que documentos extensos adquieran una inusual ventaja sobre documentos más pequeños.

La frecuencia de los términos normalizados se calcula de acuerdo a la Ecuación 3.2:

$$ntf_{i,j} = \frac{tf_{i,j}}{\text{máx}(tf_{1,1}, tf_{1,2}, ...tf_{t,j})} \tag{3.2}$$

La *frecuencia de los términos normalizada* es la relación entre la frecuencia de los términos, es decir, un término entre la frecuencia del término de máxima frecuencia, esto para cada término en el documento. Por ejemplo, si el término i aparece diez veces en el documento, y el término con mayor frecuencia aparece 100 veces, entonces $ntf_{i,j}$ es 0.1 [24].

La frecuencia inversa de los documentos (idf) es una medida de la importancia general del término y se calcula mediante la Ecuación 3.3:

$$idf_i = \log \frac{\mid D \mid}{\mid \{d : t_i \in d\} \mid} \tag{3.3}$$

donde $|D|$ es el número total de documentos en el corpus; $|\{d : t_i \in d\}|$ es el número de documentos en los que aparece el término t_i (es decir, $n_{i,j} \neq 0$). Si el término no está en el *corpus*, esto dará lugar a una división por cero. Por lo tanto, es común el uso de $1 + |\{d : t_i \in d\}|$, entonces:

$$(tf - idf)_{i,j} = tf_{i,j} \times idf_i \tag{3.4}$$

Un valor *tf-idf* alto es alcanzado por un término con alta frecuencia en el documento considerado, pero baja frecuencia en la colección total de documentos. De esta manera, el coeficiente tiende a filtrar términos comunes.

Una vez calculado el peso de cada relación verbo-sustantivo con base a la Ecuacion 3.4 se obtiene como salida una matriz como la que se ilustra en la Tabla 3.1, donde cada fila representa un vector de pesos por cada una de las palabras w. Para realizar el calculo de *tf-idf* para las relaciones verbo-sustantivo deberá existir una matriz similar a la de la Tabla 3.1 por cada documento y por todo el corpus, para poder obtener las frecuencias requeridas por las ecuaciones anteriores. El resultado final será una tabla como la de la Tabla 3.1 que contenga las frecuencias relativas y una clase asignada.

Verbos \ Sustantivos						
w_{11}	w_{12}	w_{13}	w_{14}	...	w_{1m}	$class_1$
w_{21}						$class_2$
w_{31}						$class_3$
w_{41}						$class_4$
⋮						⋮
w_{n1}	w_{n2}	w_{n3}	w_{n4}	...	w_{nm}	$class_n$

Tabla 3.1: Matriz de pesos

La representación vectorial etiquetada de la Tabla 3.1 es la entrada a una red SOM (Self-Organizing Map) que se utiliza para determinar el número de temas por documento y así ponderar el número de temas en todo el corpus. El proceso de entrenamiento de la red se describe en la siguiente sección.

3.1.2.3. Self-Organizing Map

Las ventajas de los mapas autoorganizativos radica en que son capaces de preservar la topología del espacio de los datos, proyectan datos altamente dimensionales a un esquema de representación de baja dimensión y tienen la habilidad de encontrar similitudes en los datos. Un aspecto diferenciador de la red SOM de otras muchas redes es que aprende sin supervisión, de aquí su nombre en inglés.

Se podría decir que la SOM es una "proyección no lineal" de la probabilidad de la función de densidad de los datos de entrada de alta dimensión (R^n) en una salida de dos dimensiones [35]. Al igual que cualquier método de clasificación no supervisada, también puede ser utilizado para encontrar clusters en los datos de entrada, que es el caso que nos interesa. Existen muchas versiones de la SOM ya que entre los inconvenientes que tienen las variantes de este tipo de red esta por ejemplo el seleccionar el rango de datos de entrenamiento, el número de entradas y de salidas deseado, entre otros. Por ello en esta propuesta se ha optado por utilizar el algoritmo básico, cuyos parámetros a grandes rasgos se describen a continuación.

Dado un vector de entrada x (por ejemplo $class_1$, de la Tabla 3.1) en R^n se compara con todas las m_i neuronas que forman la capa de salida con cualquier métrica. En las aplicaciones prácticas se usa habitualmente la más pequeña de las distancias euclideanas $||x - m_i||$ para definir el nodo de mejor coincidencia, representado por el subíndice c de la siguiente manera:

$$||x - m_c|| = min||x - mi|| \tag{3.5}$$

o

$$c = argmin||xmi|| \tag{3.6}$$

Por lo tanto, x se asigna o mapea al nodo c en relación con los valores de parámetros m_i. Durante el aprendizaje los nodos que están topográficamente cerca en la matriz a una cierta distancia se

activarán unos a otros para aprender de la misma entrada. Los valores iniciales de $m_i(0)$ se eligen al azar de la siguiente manera:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{3.7}$$

donde t es un número entero y representa un ciclo de ejecución y $h_{ci}(t)$ es el llamado *núcleo de vecindad*. Esta es una función definida sobre los puntos de la red. Por lo general, $h_{ic}(t) = h(||r_c - r_i||, t)$, donde $r_c \in R^2$ y $ri \in R^2$ son el radio de los vectores de los nodos de c e i, respectivamente en la matriz. Con el aumento de $||r_c - r_i||$, h_{ci} llega a 0. El peso promedio de un vector y la forma de h_{ci} define la "rigidez" de la "superficie elástica" a ser instalados en los puntos de datos del mapa final.

Para la definicion de $h_{ci}(t)$ se usa un arreglo de puntos alrededor del nodo c. Sea este conjunto de índices denotado como N_c (nótese que podemos definir $Nc = Nc(t)$ como una función de tiempo) mediante el cual $h_{ci} = \alpha(t)$ si $i \in N_c$ y $h_{ci} = 0$, si $i \notin N_c$. Este tipo de núcleo recibe el nombre de "burbuja". Otro núcleo de vecindad ampliamente aplicado se puede escribir en términos de la función de Gauss.

$$h_{ci} = \alpha(t) \times \exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}) \tag{3.8}$$

donde $\alpha(t)$ es otro valor escalar de "taza de aprendizaje", y el parametro $\sigma(t)$ define el ancho del nucleo, este ultimo corresponde al radio de Nc. Tanto $\alpha(t)$ como $\sigma(t)$ son algunas de las funciones monótonamente decrecientes y sus formas exactas no son críticas. Se ha eligido un valor de 0.2 para α y un $\sigma = 3$. Estos datos se tomaron después de la realización de diversas pruebas sobre un corpus de 250 documentos y 20909 palabras en el vocabulario.

Como se mencionaba, la red SOM puede usarse para agrupar datos, en la Figura 3.2 se muestra un ejemplo da la salida de la red para un documento. Como se puede observar, el conjunto de relaciones se agrupa en un número determinado de clases. Ese conjunto de clases representa la

cantidad de temas de las que trata el documento y el tamaño de esa submatriz de salida representa el número de relaciones aproximado que podría abarcar un tema.

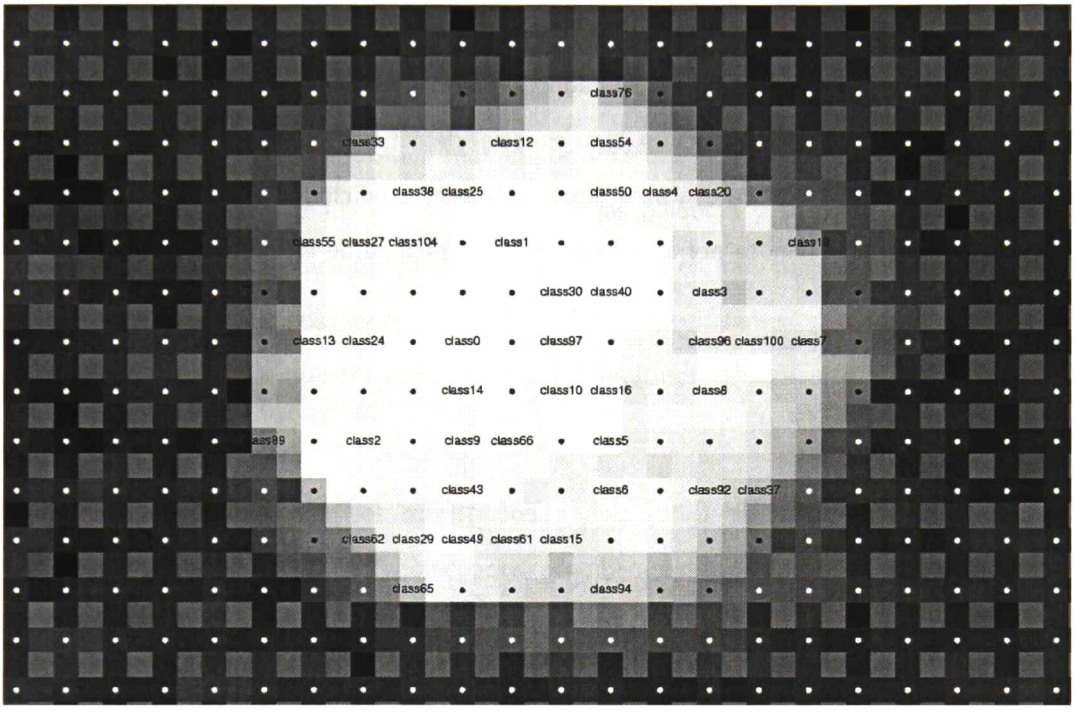


Figura 3.2: Ejemplo de una matriz de salida de la red SOM

3.1.2.4. Latent Dirichlet Allocation

Una vez que se tienen los temas y las relaciones por temas, es importante definir cuáles relaciones son las más representativas de cada tema, para ello se utiliza el algoritmo de LDA (*Latent Dirichlet Allocation*), tomando las características de la matriz de salida de la red SOM para configurar los parámetros del algoritmo LDA y utilizando la técnica de muestreo de Gibbs, ya que, como se puede leer en el trabajo de Yi Wang [63], el rendimiento de este método es comparable con otros tipos de entrenamiento LDA, pero es mejor en óptimos locales. Gibbs es un caso especial de la cadena de Markov Monte Carlo (MCMC) y produce algoritmos relativamente simples para la inferencia aproximada de grandes dimensiones, tales como los modelos de LDA.

Los métodos MCMC pueden emular grandes dimensiones de distribuciones de probabilidad $p(\vec{x})$, por el comportamiento estacionario de una cadena de Markov [27]. Esto significa que se genera una muestra para cada transición de la cadena después de que un estado estacionario de la cadena se ha alcanzado, lo que ocurre después de un llamado *"período burn-in"* que elimina la influencia de los parámetros de inicialización. Gibbs es un caso especial de MCMC, donde las dimensiones x_i de la distribución son muestreadas alternativamente una a la vez, condicionado en los valores de todas las otras dimensiones, que se denota por $\vec{x}_{\neg i}$. El algoritmo funciona de la siguiente manera:

1. Elegir la dimensión i (al azar o por permutación)
2. Muestrear x_i para $p(x_{ij}|\vec{x}_{\neg i})$.

Para construir un muestreo de Gibbs deben encontrarse los condicionales univariados (o condicionales completos) lo cual es posible mediante la Ecuación 3.9:

$$p(x_{ij}|\vec{x}_{\neg i}) = \frac{p(\vec{x})}{p(\vec{x}_{\neg i})} = \frac{p(\vec{x})}{\int p(\vec{x})dx_i} \; with \; \vec{x} = \{x_i, \vec{x}_{\neg i}\} \tag{3.9}$$

Para los modelos que contienen variables ocultas $\vec{z}$, su probabilidad posterior dada la evidencia, $p(\vec{z}|\vec{x})$, es una distribución comúnmente buscada. Con la Ecuación 3.9, la formulación general de un muestreo de Gibbs para tales modelos de variables latentes se convierte en:

$$p(z_i|\vec{z}_{\neg i}, \vec{x}) = \frac{p(\vec{z}, \vec{x})}{p(\vec{z}_{\neg i}, \vec{x})} = \frac{p(\vec{z}, \vec{x})}{\int_z p(\vec{z}, \vec{x})dz_i} \tag{3.10}$$

donde la integral cambia a una suma de variables discretas con un número suficiente de muestras $\tilde{\vec{z}}_r$, $r \in [1, R]$, así la variable latente posterior se puede aproximar mediante la Ecuación 3.11:

$$p(\vec{z}|\vec{x}) \approx \frac{1}{R}\sum_{r=1}^{R} \delta(\vec{z} - \tilde{\vec{z}}_r) \tag{3.11}$$

con la delta de Kronecker[3] $\delta(\vec{u}) = 1$ si $\vec{u} = 0$, 0 de otra manera.

Para obtener un muestreo de Gibbs para LDA se aplica el método de variables ocultas. Las variables ocultas son los temas que aparecen con las palabras del corpus $w_{m,n}$.

A continuación se describe la fórmula más importante que se utiliza para el muestreo de temas por palabras [51]. Sean $\vec{w}$ y $\vec{z}$ los vectores de todas las palabras y su tema asignado de toda la colección de datos W, la asignación de temas de una palabra en particular depende de la asignación de tema actual de todas las posiciones de palabras. Más específicamente, el tema asignado de una palabra en particular t es la muestra de la distribución multinomial siguiente.

$$p(z_i = k|\vec{z}_{\neg i}, \vec{w}) = \frac{n^{(t)}_{k,\neg i} + \beta_t}{[\sum_{v=1}^{V} n^{(v)}_k + \beta_v] - 1} \frac{n^{(t)}_{k,\neg i} + \alpha_k}{[\sum_{j=1}^{K} n^{(j)}_m + \alpha_j] - 1} \tag{3.12}$$

donde n es el número de veces que la palabra t es asignada al tema k exceptuando la asignación actual, la primera $\sum$ es el número total de palabras asignadas al tema k exceptuando la asignación actual, n es el número de palabras en el documento m asignadas al tema k exceptuando la asignación actual y la segunda $\sum$ es el número total de palabras en el documento m exceptuando la palabra actual t. En casos normales los parámetros de Dirichlet α y β son simétricos, es decir, todos los α_k son iguales y similares para β.

Después de terminar el muestreo de Gibbs se calculan dos matrices Φ y Θ de la siguiente manera.

$$\varphi_{k,t} = \frac{n^{(t)}_k + \beta_t}{\sum_{v=1}^{V} n^{(n)}_k + \beta_v} \tag{3.13}$$

[3]Delta de Kronecker es una función de dos variables, que vale 1 si son iguales, y 0 si son diferentes [13]

$$\vartheta_{m,k} = \frac{n_m^{(k)} + \alpha_k}{\sum_{j=1}^{K} n_m^{(j)} + \alpha_j} \tag{3.14}$$

donde $\Theta = \{\vartheta_m\}_{m=1}^{M}$: una matriz $M \times K$ y $\Phi = \{\varphi_k\}_{k=1}^{K}$: una matriz $K \times V$

Usando las Ecuaciones 3.12, 3.13 y 3.14 se construye el Algoritmo 2 [27]. El procedimiento sólo usa cinco estructuras de datos, las variables de conteo n_z y n_m que tienen una dimensión $M \times K$ y $K \times V$ respectivamente, las sumas de sus filas n_m y n_z con dimensión M y K, así como la variable de estado z_m con dimension W. El algoritmo de muestreo de Gibbs se ejecuta en tres períodos: inicialización, *burn-in* y muestreo.

El resultado de esta etapa es un listado de todas las relaciones verbo-sustantivo de cada documento en el que para cada documento se sabe el número de temas que abarca y las relaciones verbo-sustantivo más representativas de cada tema.

3.1.3 Etapa III: Recuperación de significados

En los actuales modelos y sistemas de recuperación de información, sólo cuando las palabras de la consulta aparecen en el documento, el documento puede ser recuperable. Sin embargo, el mismo concepto por lo general tiene varios métodos de expresión en lenguaje natural. Como resultado, es probable que los documentos que si estén relacionados se puedan omitir debido a la diferencia en su expresión. Este tipo de problema es una de las razones más importantes que influyen de la exactitud de recuperación de información [60].

Dado que un usuario no puede tener un conocimiento completo del dominio y a menudo no puede especificar las palabras clave adecuadas y exactas para una consulta válida, hemos optado por agregar una fase de "expansión semántica", un proceso que ha sido utilizado exitosamente en trabajos como el de [74] y [19] en el desarrollo de ontologías para ampliar información y en los trabajos de [54] y [14] en "expansión de consultas" para ampliar las zonas de búsqueda.

Algoritmo 2 Algoritmo de muestreo de Gibbs para latent Dirichlet allocation

Inicialización. Poner a cero todas las variables de conteo $n_m^{(k)}$, n_m, $n_k^{(t)}$, n_k
para todos los documentos $m \in [1, M]$ **hacer**
  para todas las palabras $n \in [1, N_m]$ en el documento m **hacer**
    muestrear el indice de temas $z_{m,n} = k \sim$ Mult (1/K)
    incrementar contador de temas por documento $n_m^{(k)} + 1$
    incrementar suma de temas por documento $n_m + 1$
    incrementar contador de temas por término $n_k^{(t)} + 1$
    incrementar suma de temas por término $n_k + 1$
  fin para
fin para
Gibbs en período burn-in y el período de muestreo
mientras no ha terminado **hacer**
  para todos los documentos $m \in [1, M]$ **hacer**
    para todas las palabras $n \in [1, N_m]$ en el documento m **hacer**
      para la asignación actual de k para un término t de la palabra $w_{m,n}$ **hacer**
        decrementar contador y suma: $n_m^{(k)} - 1$; $n_m - 1$; $n_k^{(t)} - 1$; $n_k - 1$
      fin para
      muestreo multinomial acc. para la ecuación. 3.12 (decrementar desde el paso anterior):
      muestrear el indice de temas $\tilde{k} \sim p(z_i | \vec{z}_{\neg i}, \vec{w})$
      usar la nueva asignación de $z_{m,n}$ para un termino t de la palabra $w_{m,n}$:
      incrementar contador y sumas:$n_m^{(k)} + 1$; $n_m + 1$; $n_k^{(t)} + 1$; $n_k + 1$
    fin para
  fin para
  comprobar la convergencia y la lectura de los parámetros
  si converge y L iteraciones de muestreo desde la ultima lectura de salida **entonces**
    los diferentes parámetros lectura de salida se promedian.
    lectura de los parámetros establecidos Φ de acuerdo a la ecuación. 3.13
    lectura de los parámetros establecidos Θ de acuerdo a la ecuación. 3.14
  fin si
fin mientras

Para este proceso se requiere encontrar el sentido de cada término significativo del texto en WordNet. Para ello se emplean los sinónimos, hipónimos e hiperónimos de esta base de datos. Recordemos que tenemos dos conjuntos de términos, uno de verbos y otro de sustantivos. Aunque en pasos anteriores se ha trabajado con ellos en forma de una sola relación, en esta etapa se procesan de forma independiente ya que WordNet maneja estos dos grupos de forma separada,

así que las consultas se deben hacer considerando la etiqueta de verbo o sustantivo respectiva para cada término. La finalidad de este proceso es ayudar a ampliar la búsqueda adicionando términos a los ya encontrados en el texto. En la Figura 3.3 se puede observar que del verbo *domesticate* se derivan otros verbos pero ningún sustantivo.

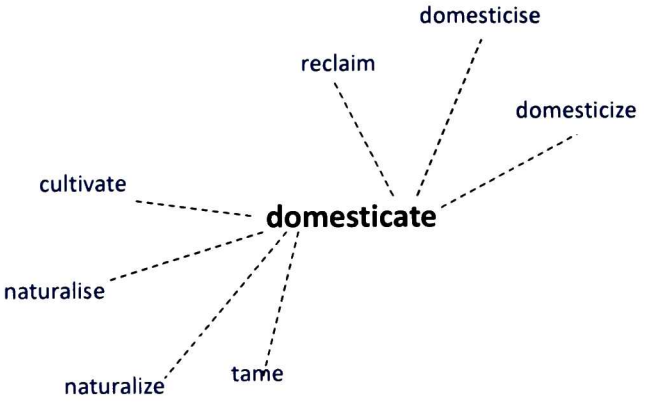


Figura 3.3: Ejemplo de recuperación de significados

3.1.4 Etapa IV: Obtención de relaciones

Con el conjunto de palabras que son retornadas de WordNet se forma un grafo en el que cada palabra representa un nodo, un nodo puede tener varias palabras si las palabras comparten un mismo significado. Esto podría parecer confuso para el caso de una relación de sinonimia, pero visto desde el punto de vista de la estructura de datos, (Figura 3.5), que se explica más adelante, la información de la relación de sinonimia está en el nodo. El objetivo de que un nodo tenga más de un término es por cuestiones de espacio, será indistinto se maneja un nodo por cada término.

Cada vez que recuperamos los términos relacionados a una palabra, es decir los sinónimos, hipónimos e hiperónimos, se van creando los enlaces entre ellos. Tomando en cuenta que no deben existir términos repetidos en el grafo, cada término tiene una clave asignada tomada de WordNet, si el término no tiene un equivalente en WordNet entonces se le asigna un número negativo. Estos enlaces

permiten establecer relaciones entre diferentes términos considerando la relación de palabras con sentido similar, es decir, qué tan fuerte o débil es la relación entre cada término de los documentos.

En la Figura 3.4 podemos ver un pequeño ejemplo de estas características para la palabra *"dog"*. Esta palabra tiene un hiperónimo que es *"canine"* y dos hipónimos que son *"puppy"* y *"doggy"*, en este ejemplo también podemos ver las claves que WordNet ha asignano a cada término. Dado que el ejemplo es muy sencillo también podemos ver la descendencia, es decir la relación indirecta entre *"canine"* y *"puppy"* por ejemplo.

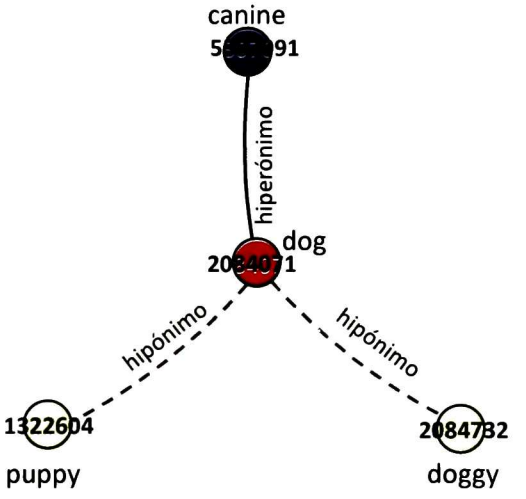


Figura 3.4: Ejemplo de relaciones semánticas

También es posible agregar más información a los nodos, es una de las características de este tipo de estructura. En la sección 3.2 veremos que son necesarios ciertos datos adicionales para poder realizar las consultas, en dicha sección se describe de forma más detallada la estructura del grafo.

Teniendo el grafo de términos relacionados se genera un meta-grafo (grafo de grafos) de todos los documentos en conjunto. Esto se logra cuando ya se sabe qué nodos corresponden a un determinado documento.

3.1.5 Etapa V: Búsqueda en el meta-grafo

Por último, y para demostrar que esta estructura sirve no sólo como medio de representación semántica, se aplican los algoritmos del *cálculo local* y *cálculo global* para la búsqueda de documentos en el grafo. La descripción y uso de estos algoritmos se detalla en la sección 3.3, el primer tema de esta sección se refiere a la forma en la que se estructura la información en el grafo, también trata sobre el esquema general de las búsquedas, las dos perspectivas del grafo (la de términos y la de documentos) así como la importancia que tiene cada consulta.

3.2 Estructura de datos para la representación de información

Como ya se mencionó, el tipo de representación que se utiliza para los datos es un grafo no dirigido que puede ser conexo o disconexo. La Figura 3.5 ilustra un grafo formado por los subgrafos de tres palabras; es de tipo disconexo ya que los tres subgrafos no se encuentran enlazados hasta este momento. Conforme se agregan más términos habrá un mayor número de conexiones, permitiendo que se convierta en un grafo conexo. Sin embargo, trabajar con un grafo disconexo, es lo más probable. Esto ocurre porque pueden llegar a existir términos muy especializados dentro de un documento que no se encuentran en la base de datos WordNet. Al no poder encontrar qué relación de sinonimia, hiperonimia o hiponimia puede tener un término con respecto a otros en el mismo documento, no será posible enlazarlo, convirtiéndose así en un subgrafo o nodo aislado. Sin embargo, esto no representa un problema para las consultas, ya que la aplicación desarrollada para la búsqueda en el grafo considera también los nodos aislados.

En la Figura 3.5 también se puede observar la información que contiene cada nodo del grafo. Por ejemplo: el primer dato indica su clave, si se trata de un verbo, como en la figura, o un sustantivo; a qué palabra o palabras se hace referencia en ese nodo. La información del nodo también nos dice en qué documentos podemos encontrar esa palabra y si se trata de un sinónimo, hiperónimo o hipónimo

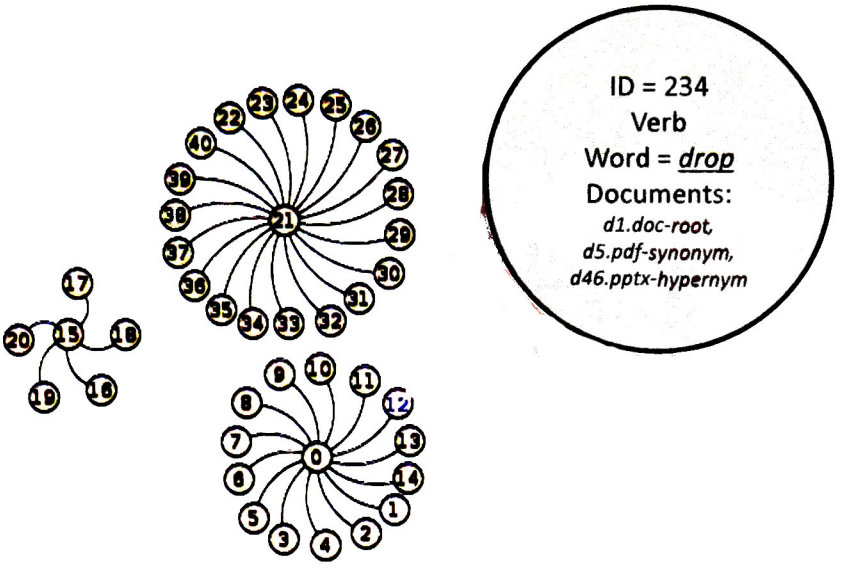


Figura 3.5: Información que contienen los nodos del grafo.

derivado de algún término contenido en los documentos que se mencionan en el nodo. Todos estos elementos se utilizan para realizar las búsquedas de información cuando se realiza una consulta.

La Figura 3.6 ilustra un metagrafo como ejemplo de la agrupación de los términos por documento, donde cada círculo en rojo es un término y los círculos exteriores son documentos. En este pequeño ejemplo se puede observar cómo la información se traslapa, es decir un término puede pertenecer a más de un documento a la vez.

3.3 Búsqueda en el meta-grafo

La búsqueda consiste en dos evaluaciones, primero realizar un cálculo local para encontrar los términos en los documentos y un segundo cálculo global para encontrar documentos de temas afines. Esto se explica con más detalle a continuación.

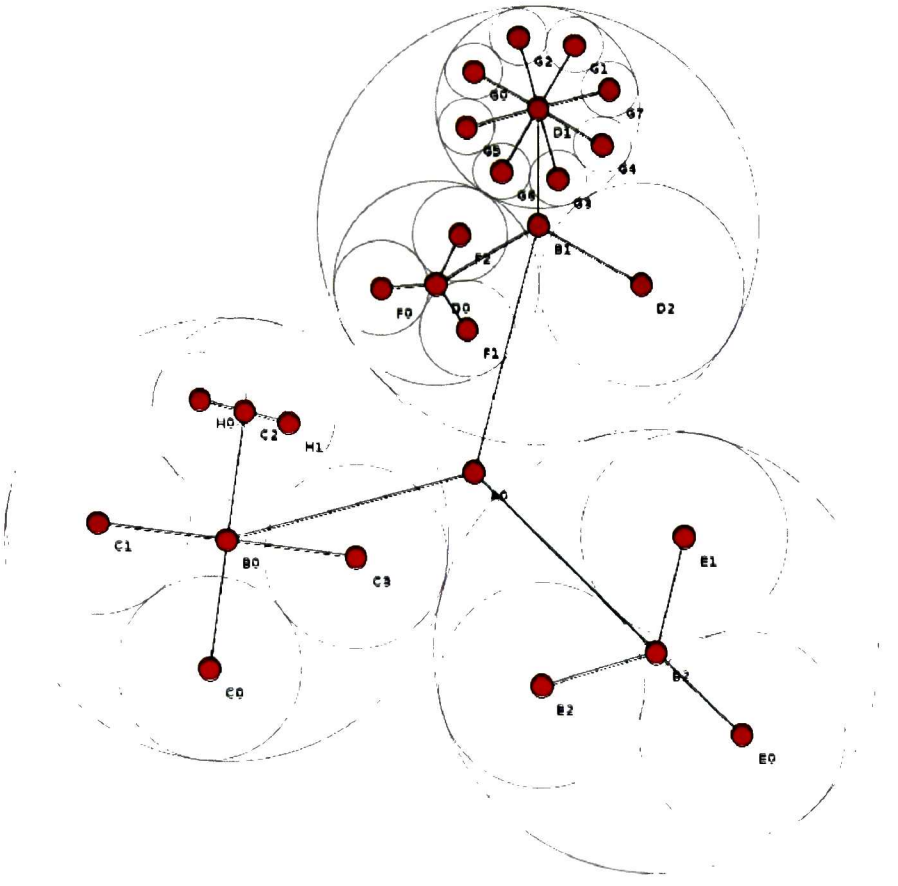


Figura 3.6: Ejemplo de agrupación de términos por documento.

3.3.1 Cálculo Local

Con el meta-grafo ya se tiene una forma de representar los documentos y conocer la relación entre ellos. El siguiente paso es poder acceder a esa información. En esta sección se presentan un conjunto de fórmulas, las cuales permiten a un usuario realizar una consulta y obtener como resultado una serie de documentos que cumplan sintáctica y semánticamente con la información que el usuario ha proporcionado como entrada.

Al introducir una consulta, ésta se trasforma a un vector de entrada, llamado "vector de búsqueda", representado de la siguiente manera.

$$
\begin{aligned}
q = \{q_1, q_2, \ldots q_m\} \equiv [1{,}0, 1{,}0, \ldots 1{,}0] \\
d_1 = \{q_1, q_2, \ldots q_m\} \\
d_2 = \{q_1, q_2, \ldots q_m\} \\
\vdots \\
d_n = \{q_1, q_2, \ldots q_m\}
\end{aligned}
\tag{3.15}
$$

$$q \cdot d_i = \|q\| \, \|d_i\| \cos\theta$$

donde $q_1 ... q_m$ son los lexemas de las palabras introducidas en la búsqueda. El lexema es una palabra que constituye la unidad mínima con significado léxico.

Por la forma en la que se ha construido el grafo, se consideran tres tipos de lexemas: verbos, sustantivos y sustantivos compuestos. Este vector se compara contra los "vectores de correspondencia de los documentos", donde cada vector d_j es el *j-ésimo* vector formado por la coincidencia de los lemas encontrados en el documento j. Para ello se observa si el término q_i existe o no en cada documento del grafo. Un caso especial es el de los sustantivos compuestos, que se forman de dos sustantivos, sino se encuentra el sustantivo compuesto se busca cada sustantivo por separado. En este caso puede ocurrir que cada sustantivo individual pueda acompañar o no a un sustantivo distinto.

De acuerdo a la experimentación realizada, cuando se encuentra un término lematizado en el documento se deben considerar las siguientes ponderaciones:

- **1.0** si la palabra encontrada está en el documento
- **0.9** si la palabra encontrada en el grafo es un sinónimo de alguna palabra que se encuentre en el documento

- **0.75** si la palabra encontrada en el grafo es un hiperónimo de una palabra que se encuentre en el documento

- **0.60** si la palabra encontrada en el grafo es un hipónimo de una palabra que se encuentre en el documento

Las medidas de similitud son una de las herramientas más importantes para saber el grado de similitud entre objetos. Funciones que expresan el grado de similitud de los elementos o conjuntos se utilizan en la antropología física, taxonomía numérica, ecología, recuperación de la información, psicología, análisis de citas, y la clasificación automática. De hecho, el grado de similitud o diferencia entre los objetos de es un problema de investigación abierto.

En la expansión de consultas, varias medidas de similitud término-término basadas en la dirección de vectores se han sugerido para seleccionar los términos de búsqueda adicionales. Las medidas de similitud Jaccard, Dice y coseno se utilizan a menudo para este propósito [70]. Hablando específicamente del problema que intentamos resolver en esta sección con el desarrollo de una representación vectorial para los documentos recuperados, la medida de similitud de coseno es considerada la mejor para la búsqueda de similitud de documentos [59], ya que como su nombre lo indica se basa en la semejanza y no en la coincidencia exacta.

La forma de calcular qué tan similar es el vector de búsqueda contra el vector de documentos utilizando la medida de similitud de coseno es obteniendo el coseno del ángulo entre vectores [43], como se muestra en la Ecuación 3.16

$$\cos(\theta) = \frac{q \cdot d_i}{\|q\|\|d_i\|} = \frac{\sum_{j=1}^{m} q_j \times d_{i_j}}{\sqrt{\sum_{j=1}^{m} (q_j)^2} \times \sqrt{\sum_{j=1}^{m} (d_{i_j})^2}} \tag{3.16}$$

La ecuación anterior considera a cada vector como una línea representada en un plano cartesiano y calcula la separación entre estas dos líneas. El resultado de la función coseno es igual a 1 cuando el ángulo es 0, y es inferior a 1 cuando el ángulo es cualquier otro valor. Esta función sirve para determinar si dos vectores apuntan en aproximadamente la misma dirección, es decir si son parecidos. Dados dos vectores de atributos q y d, la similitud del coseno θ se representa mediante un producto escalar y magnitud, como en la Ecuación 3.16, donde el numerador representa el producto escalar de los vectores q_j y d_{i_j}, mientras que el denominador es el producto de su longitud euclidiana.

3.3.2 Cálculo Global

Podrían existir verbos o sustantivos muy comunes en el grafo que podrían producir que el cálculo local diera mayor importancia a términos más comunes, cuya frecuencia es mayor. Para evitar este problema se utiliza el algoritmo de *PageRank (PR)* como medida de cálculo global.

En los últimos años se han desarrollado y probado varios métodos de recuperación de información centrados en la estructura de enlaces que mejoran significativamente el rendimiento de las búsquedas en Internet, el algoritmo de PageRank y HITS son ejemplos de ello.

PageRank es considerada una de las primeras técnicas de análisis basada en el enlace que aumentan el rendimiento de recuperación de información en la Web, dado que PageRank, expresa la ïmportancia", o rango, de la página Web en toda la Web [52].

Desde el punto de vista matemático el algoritmo de PageRank es un modelo basado en grafos, y la semejanza que tiene la Web vista como un grafo no dirigido [47], al igual que la representación gráfica que se propone en este trabajo sumado a los importantes resultados que ha tenido este algoritmo en la búsqueda y ponderación de paginas [62], [31], [1], [26], entre muchos otros, lo han convertido en el modelo elegido como una ponderación adecuada para obtener una medida de importancia global para los documentos.

Para lograr lo anterior, se usa un grafo con las conexiones de las relaciones verbo-sustantivo entre documentos como el que se muestra en la Figura 3.7.

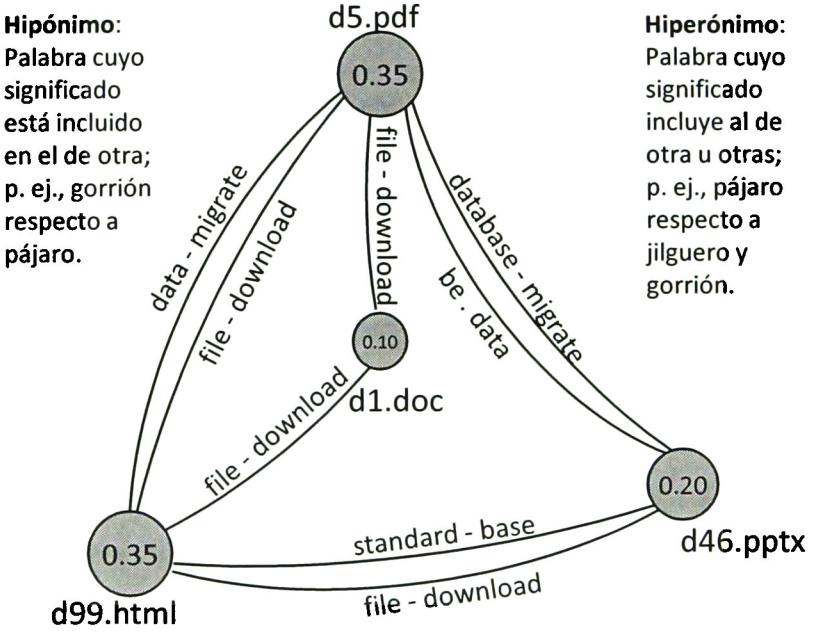


Figura 3.7: Ejemplo de representación gráfica de 5 relaciones que enlazan 4 documentos.

Estas relaciones se obtienen mediante el algoritmo LDA y representan las relaciones más significativas de cada conjunto de temas correspondiente a un documento. Para este cálculo no se consideran todos los documentos, sólo aquellos que obtuvieron la puntuación más alta y que superan la media en el cálculo local. A partir de este pequeño conjunto se busca con qué documentos se relacionan y cuántas conexiones tienen entre ellos. La idea del cálculo global es obtener el peso de un documento con respecto a los demás. Así, documentos de temas afines estarán más relacionados y tendrán un mayor peso. Es en este paso cuando los nodos aislados (que son términos más especializados dentro del documento) cobran mayor importancia. El algoritmo inicial del PageRank se muestra en la Ecuación 3.17.

$$PR(D) = (1 - c) + c * \sum_{i=1}^{n} \frac{PR(d_i)}{N(d_i)} \tag{3.17}$$

donde $PR(D)$ es el valor del PageRank para el documento D, $PR(d_i)$ significa el valor del PageRank para el documento *i-ésimo* y un enlace de d_1 a D, $N(d_i)$ es el número de otro documento (nodo) al que se enlaza, c es un valor entre 0 y 1 (por lo regular equivale a 0.85) y n es el número de documentos a los cuales se enlaza D. El valor total asignado a cada documento es el resultado de la suma de los pesos locales determinados por la aparición de términos en el grafo y el número de conexiones a otros documentos, como se muestra en la Ecuación 3.18.

$$\text{similarity} = \cos(\theta) + PR(A) \tag{3.18}$$

Desde la perspectiva del calculo global, también podría ocurrir que todos o la mayoría de los documentos estuvieran conectados. Dada esta consideración adicional, no se pueden tomar en cuenta todos los documentos que se retornen del calculo global.

De acuerdo al resultado de la experimentacion se ha determinado que el número de documentos recuperados será como lo define la Ecuación 3.19.

$$documentos\ recuperados = rcl/n \tag{3.19}$$

así se obtendrá el número ideal de documentos a retornar en una consulta, donde n es el número de lemas en una consulta y rcl corresponde a la cantidad de documentos recuperados por el cálculo local. Dado que para los rcl el primer grupo corresponde a los documentos que contienen más lemas asociados a la consulta. En un caso ideal estos serían los n lemas, el siguiente grupo tendría $n - 1$ lemas, decrementándose el valor de n hasta llegar al grupo que contiene solamente un lema asociado.

Entendemos que el grupo más importante es el primero, dado que tiene más lemas asociados.

Sin embargo, este grupo es sumamente reducido, ya que disminuye en razón del número de lemas. Dadas estas consideraciones y para hacer un reparto más equitativo se divide el grupo de ganadores en los n lemas como se mostró en la ecuación 3.18. Así se tomarán los documentos del primero y parte del segundo o tercer grupo según sea el caso, considerando que los ganadores del cálculo global no son obligatoriamente los mismos que los ganadores del cálculo local.

3.4 Implementación en software

En lo referente al software, el lenguaje que se propuso para el desarrollo del prototipo fue Java mediante el entorno de desarrollo NetBeans 6.8 bajo la plataforma Linux Ubuntu 10.10-Maverick Meerkat. Se ha considerado este lenguaje debido a que facilita la integración de algunas herramientas existentes que fueron esenciales para el desarrollo de la aplicación y para la integración de la aplicación en otras herramientas.

El prototipo desarrollado para la metodología propuesta ha permitido implementar y evaluar las ecuaciones 3.15, 3.16, 3.17, 3.18 y 3.19 como método de búsqueda y recuperación de información. Los resultados de la evaluación de esta implementación se describen en el siguiente capítulo.

Se han utilizado las siguiente herramientas para el desarrollo de la aplicación de prueba, la mayoría de ellas están codificadas en Java: Apache Tika, The Stanford Parser, Stanford Log-linear Part-Of-Speech Tagger, JGibbsLDA, SOM PAK codificado en C, API RiTa, WordNet y JUNG Framework.

Una superposición de las herramienta utilizadas y los bloques que integran la solución de la representación gráfica, se describe a continuación, y puede verse más claramente en el diagrama de la Figura 3.8. Los bloques en gris representan los módulos en los cuales se tomado de otros autores.

- **Extracción de texto**: Apache Tika[4].

[4]Disponible en `http://tika.apache.org/`, accesado en julio 2011

- **Extracción de relaciones**: The Stanford Parser[5].
- **Lematización**: Stanford Log-linear Part-Of-Speech Tagger[6].
- **Representación vectorial**: Se desarrolló una aplicación en Java implementando las ecuaciones 3.2, 3.3 y 3.4.
- **LDA**: JGibbsLDA: A Java and Gibbs Sampling based Implementation of Latent Dirichlet Allocation (LDA)[7].
- **SOM**: SOM PAK: The Self-Organizing Map program package[8].
- **Acceso a Wordnet**: API RiTa.WordNet[9].
- **Visualización del grafo**: JUNG — The Java Universal Network/Graph Framework[10].

Para el caso de la extracción de texto se desarrolló un programa en Java que explorara todos los archivos del directorio de entrada y recolectara sólo aquellos que tuvieran una de las extensiones de archivos visto en sección 3.1.1 de la metodología. Una vez hecho esto sólo se requirió hacer uso de la función "tika.parser" de Apache Tika para realizar el proceso de extracción. Como nota adicional, algunos archivos pueden requerir la eliminación de caracteres extraños, pero ello sólo se requiere eliminar estos caracteres utilizando expresiones regulares, se sugiere conservar sólo letras y signos de puntuación.

En el caso de Stanford Parser es muy similar al anterior, se desarrolló un programa en Java que procesa independientemente cada oración de cada archivo de entrada, creando un

[5]Disponible en `http://nlp.stanford.edu/software/lex-parser.shtml#Download`, accesado en julio 2011
[6]Disponible en `http://nlp.stanford.edu/software/tagger.shtml`, accesado en julio 2011
[7]Disponible en `http://sourceforge.net/projects/jgibblda/`, accesado en julio 2011
[8]Disponible en `http://www.cis.hut.fi/research/som_pak/`, accesado en julio 2011
[9]Disponible en `http://www.rednoise.org/rita/wordnet/documentation/index.htm`, accesado en julio 2011
[10]Disponible en `http://jung.sourceforge.net/download.html`, accesado en julio 2011

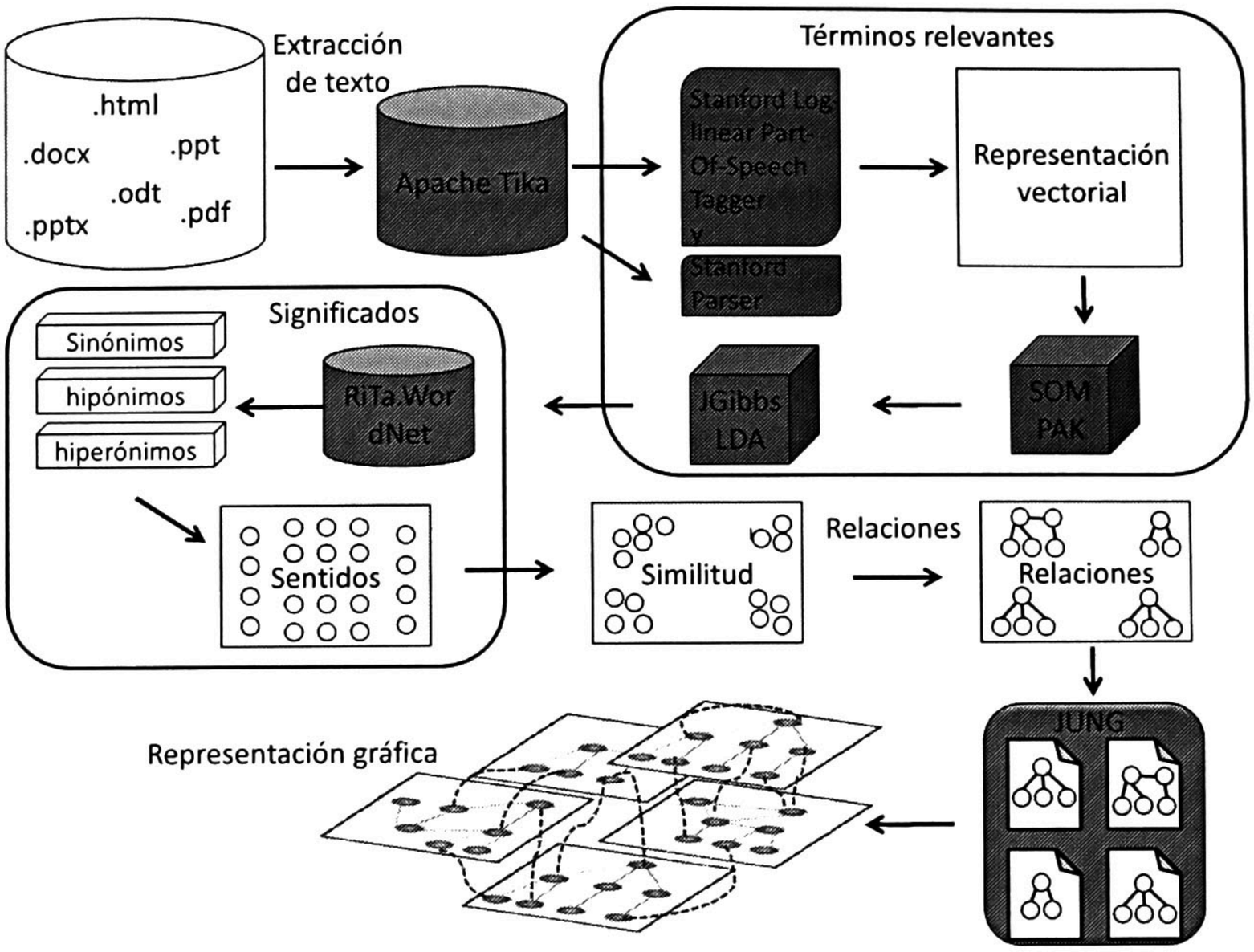


Figura 3.8: Herramienta utilizadas en los bloques que integran la solución de la representación gráfica.

árbol gramatical para cada oración con las funciones "LexicalizedParser", "wordsAndTags" y "typedDependenciesCollapsed", seleccionando sólo los verbos y sustantivos que concuerdan con la funcion "lemmatiza" del Stanford Log-linear Part-Of-Speech Tagger. Al mismo tiempo que se lleva este análisis se realiza en conteo de las veces que aparecen las relaciones entre los documentos.

Para la representación vectorial se implementaron en Java las ecuaciones 3.2, 3.3 y 3.4 que obtienen las frecuencias de las relaciones a partir de los conteos de ocurrencia hachos por el parser.

Para el caso de SOM Pack se creó un código en C para invocar a las funciones "randinit", "vsom", "vcal", "visual" y opcionalmente "umat" para almacenar la información de salida en un archivo que representa los parámetros para LDA.

Para el caso de LDA, sólo se requirió una modificación para procesar los documentos como archivos independientes y no en un solo archivo. Aunque este proceso es innecesario, no altera

en lo absoluto el desarrollo de algoritmo LDA. Para ello se empleó la función "estimate" del API JgibbsLDA.

En el caso del API RiTa, se utilizaron las funciones "getSenseIds", "getAllSynonyms", "getAllHypernyms" y "getAllHyponyms" para obtener los sinónimos, hiperónimos e hipónimos respectivamente. La formación del grafo se describió en las sección 3.2 y se puede apreciar más fácilmente en las figuras 3.4, 3.5, 3.6, 3.7, 4.9 y 4.9.

Uno de los usos del framework JUNG fue mostrar visualmente el grafo, aunque este paso puede descartarse. El uso realmente relevante de este framework fue el algoritmo de PageRank tomado de la función "algorithms.scoring.PageRank".

Para las búsquedas de también se desarrolló código en Java que implementa las ecuaciones 3.15, 3.16, 3.17, 3.18 y 3.19. En el apéndice A se muestra el diagrama de clases para esta implementación.

Un breve ejemplo del desarrollo de lo que se realiza en estos pasos se da en el capítulo 4. A lo largo de esta tesis se dan los detalles de los algoritmos y sus parámetros para que la comprobación del método pueda ser recreada sin depender de una herramienta en específico. El listado de las herramientas que se muestra en esta sección son una sugerencia que facilitaría el desarrollo de la aplicación, pero existe código disponible de otros autores para los mismos algoritmos, de tal modo que el desarrollo de la aplicación se deja a criterio del lector.

3.5 Resumen

En este capítulo se han planteado tres contribuciones para la búsqueda y recuperación de información. Como primer punto se ha proporcionado un enfoque para una representación gráfica de los documentos a través de la serie de pasos detallados en la sección "Metodología para la construcción del grafo semántico". En estas etapas para la construcción del grafo se puede observar

cómo la extracción de características esenciales de un conjunto de documentos es una tarea bastante complicada, sobretodo si se trata de un proceso que pretende ser totalmente independiente del usuario. Sin embargo realizar este proceso de forma manual le tomaría demasiado tiempo a una persona.

Como segundo punto se ha proporcionado un método de búsqueda y recuperación de información sobre la estructura de datos propuesta, con la que se ha podido sacar provecho a los enlaces semánticos que se obtienen gracias a la representación por grafos.

Como último punto se propuso una fórmula para reducir la cantidad de documentos retornados por una consulta.

Para finalizar este capítulo se presentó un conjunto de herramientas para la implementación en software de la metodología propuesta.

3.6 Conclusiones

Tomando como referencia la metodología para la construcción del grafo, podemos concluir que el establecimiento de las relaciones semánticas es sencillo y fácil de entender para un usuario dado que es similar al proceso que realiza la mente humana. Además de ser una estructura semántica fácil de entender, le da una ventaja a este método por sobre otras representaciones más complejas.

Podemos ver que en cada uno de los pasos de la metodología se intenta dar solución a diferentes problemas de la búsqueda y representación de información, concluyendo entonces que si este proceso se realizara de forma manual requeriría demasiado tiempo para un ser humano realizarlo por completo dado un grupo significativamente grande de datos. Esta es otra de las ventajas que podemos ver en esta propuesta.

En lo referente a las propuestas de los algoritmos búsqueda, si observamos detalladamente, estas fórmulas pueden ser utilizadas en la "expansión de consultas", en las que el proceso de construcción del grafo podría llevarse acabo a la inversa, es decir construyendo un grafo para la consulta. Esto se concluye a raíz de la idea de que en algunos casos no se pudiera indexar la información del usuario, basados en esta propuesta estos 2 algoritmos de búsqueda pueden ser implementados de forma independiente.

Al igual que en el caso anterior la fórmula de recuperación de información es un buen punto de partida ya que considera la relación consulta-contenido. Sin embargo una de las posibles desventajas que tendría esta ecuación es que reduce al mínimo la cantidad de documentos a mostrar a un usuario.

4

Pruebas y resultados

En este capítulo se presenta la evaluación y resultados obtenidos de la puesta en práctica de la metodología descrita en el capítulo anterior. Se presentan los resultados de experimentos con dos grupos de documentos, uno ampliamente utilizado en la literatura y uno propio.

4.1 Introducción

El objetivo de este capítulo es describir el proceso de evaluación de la representación mediante grafos que se propone y demostrar la eficiencia del uso de esta propuesta para recuperación de información. La primera parte del capítulo se centra en el marco de trabajo para la evaluación de la aplicación desarrollada. Para mostrar la propuesta se presentan experimentos y detalles de configuración de éstos. Posteriormente se describen las características de los corpus de prueba utilizados y las métricas de evaluación aplicadas para medir el desempeño de la propuesta. Finalmente se presenta la evaluación del prototipo desarrollado comparando los resultados obtenidos con la aplicación de escritorio de *Google* (*Google Desktop*) y la aplicación semántica *SemanticVectors*.

4.2 Casos de prueba

En el primer caso se realizaron pruebas con un conjunto de documentos en formato PDF tomados del sitio *springerlink.com*, los cuales fueron clasificados manualmente considerando un escenario de cinco temas con 50 archivos por tema sumadno un total de 250 documentos. A continuación se presenta una lista de los temas con los cuales se desarrollaron las pruebas, éstos pertenecen al área de Ciencias Computacionales. Se eligió esta área ya que involucra términos muy especializados, el uso de términos técnicos es considerado como los peores casos para la aplicación de la propuesta ya que la base de datos WordNet, que se utiliza para la extracción de términos, contiene sólo términos generales no especializados del idioma inglés:

- Algorithmica
- Knowledge and Information Systems
- Multimedia Tools and Applications
- Software quality
- Supercomputing

Cada documento de este corpus tiene en promedio 16 páginas. De las cuales se han extraído 27434 términos relevantes.

El segundo caso es el corpus de 20 Newsgroups[1]. 20 Newsgroups es una colección de aproximadamente 20.000 documentos, en particiones (casi) iguales de 20 diferentes grupos de noticias. Este fue originalmente recolectado por Ken Lang, probablemente para su artículo *Newsweeder: Learning to filter netnews* [39], aunque no menciona explícitamente esta colección. La colección de 20 Newsgroups se ha convertido en una colección de datos popular para experimentos

[1]Disponible en `http://people.csail.mit.edu/jrennie/20Newsgroups/`, accesado en noviembre del 2011

en aplicaciones de texto para técnicas de aprendizaje automático, tales como la clasificación de texto y el agrupamiento de texto.

Para los 20 diferentes grupos en los que está organizada la colección, cada grupo corresponde a un tema diferente. Algunos de los grupos de noticias están estrechamente relacionadas entre sí (por ejemplo comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), mientras que otros están poco relacionados (por ejemplo, misc.forsale / soc.religion.christian). La Tabla 4.1 muestra una lista de los 20 grupos de noticias, dividido (aproximadamente), de acuerdo a cada temática:

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

Tabla 4.1: Los grupos de noticias dividido por tema del corpus 20 Newsgroup

Para la experimentación con el corpus 20 Newsgroups se tomaron cuatro temas que suman un total de 3696 documentos:

- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- sci.crypt
- talk.religion.misc

De los cuales se puede observar que sólo dos están estrechamente relacionados. El objetivo de usar grupos de datos que tienen poca intersección es ver si la aplicación es capaz de hacer esa misma distinción entre los grupos.

4.3 Validación experimental

En esta sección se presenta una descripción del equipo de prueba utilizado para la fase de pruebas y los resultados parciales que se van obteniendo en cada una de las etapas de la metodología.

4.3.1 Infraestructura de prueba

Se contó con una computadora de escritorio con las siguientes características:

- Procesador Intel Core 2 Duo a 2.93GHz
- 8 Gb de memoria RAM
- 500 Gb de disco duro

4.3.2 Salidas por etapas

Con el objetivo de ilustrar la forma en la que se obtiene y organiza la información se tomaron tres archivos como ejemplo del corpus de Springer del área de aplicaciones multimedia y que están relacionados con temas de *videojuegos* y *dispositivos moviles*. A continuación se describen los resultados obtenidos en cada una de las etapas de la metodología.

Extracción de texto

Después de descargar los archivos en una carpeta de trabajo se ejecuta el Stanford Parser para el etiquetado de todos los términos. En este paso no se deben quitar los stopwords (palabras con poca aportación) ya que son tomadas en cuenta por el parser para la determinación de las etiquetas para cada palabra. Para ayudar al procesamiento del parser, cada oración se divide por un salto de línea. Se aceptan sólo letras, números, saltos de línea, tabuladores, signos de puntuación y algunos

caracteres especiales (-.,;@¿?!¡><()[], _%$&#=|).

Las figuras 4.1, 4.2 y 4.3 muestran el resumen y las conclusiones respectivamente de cada uno de los 3 archivos de ejemplo (para efecto de ilustrar el contenido de los artículos).

Abstract Treasure is a pervasive game playing in the context of people's daily living environments. Unlike previous pervasive games that are based on the predefined contents and proprietary devices, Treasure exploits the "design-in-play" concept to enhance the variability of a game in mixed-reality environments. Dynamic and personalized role design and allocation by players is enabled by exploring local smart objects as game props. The variability of the game is also enhanced by several other aspects, such as user-oriented context-aware action setting and playing environment redeployment. The effectiveness of the "design-in-play" concept is validated through a user study, where 15 subjects were recruited to play and author the trial game.

...

6 Conclusion

Treasure represents our early efforts to increase the variability of pervasive games in mixed-reality environments. The "design-in-play" principle, which allows end users to create or alter contents before each game play, is proposed to achieve this. To support dynamic and personalized role design, we explore the numerous smart everyday objects in smart homes as props. Treasure further allows players to configure the context-aware actions of the roles they created and reorganize the physical objects in the gaming environment. The evaluation results from an initial user study reveal that diverse gaming experiences can be generated and better user experience can be attained by allowing users to configure different elements of a meta-pervasive-game.

Based on the feedback from the subjects, we intend to make the authoring kit more user-friendly by introducing more graphical elements. In view of the fact that user content sharing has been a trend in the social networking era, we plan to create a general platform that enhances user cooperation and content sharing in the development of indoor pervasive games. Mechanisms will also be provided that allow users to create simple games and share their experiences with their peers.

Figura 4.1: Resumen y conclusiones del Documento 1.

Abstract With advances in broadcasting technologies, people are now able to watch videos on devices such as televisions, computers, and mobile phones. Scalable video provides video bitstreams of different size under different transmission bandwidths. In this paper, a semantic scalability scheme with four levels is proposed, and tennis videos are used as examples in experiments to test the scheme. Rather than detecting shot categories to determine suitable scaling options for Scalable Video Coding (SVC) as in previous studies, the proposed method analyzes a video, transmits video content according to semantic priority, and reintegrates the extracted contents in the receiver. The purpose of the lower bitstream size in the proposed method is to discard video content of low semantic importance instead of decreasing the video quality to reduce the video bitstream. The experimental results show that visual quality is still maintained in our method despite reducing the bitstream size. Further, in a user study, we show that evaluators identify the visual quality as more acceptable and the video information as clearer than those of SVC. Finally, we suggest that the proposed scalability scheme in the semantic domain, which provides a new dimension for scaling videos, can be extended to various video categories.

...

5 Conclusions and extensions

To provide videos with variable bitstream sizes, we propose four levels of scalability in the semantic domain. Without compromising video quality to reduce the bitrates, a lower bitstream size is achieved by discarding video content of low semantic importance. Experimental results show that video bitrates can be effectively reduced by re-using the background scenes, discarding some video clips, reducing the amount of foreground objects, and replacing the video with animation, respectively. Subjective evaluations reveal that the visual quality of videos compressed by our proposed method is better than that of videos compressed by SVC at low bitrates. In addition, understanding is clearer when watching videos compressed by the proposed method than those compressed by SVC at low bitrates. To the best of our knowledge, the proposed scalability scheme in the semantic domain is different from previously studied schemes and provides a new dimension for scalable video coding.

Although we only used tennis videos as examples in this paper, semantic scalability can be extended to further video categories such as home videos and football videos. To extend the concept of a Video Unit to such videos, play clips (in tennis) can be replaced with clips attracting more visual attention [9] and shoot clips, respectively. Video analysis is a key step to achieve semantic scalability; the process of foreground segmentation by projecting video frames to the sprite plane can be applied to various video categories [7]. To extract video information, camera motion can be used as a clue to annotate home videos [1], and previous methods of extracting football and player trajectories are available [4, 17]. With video information and content thus extracted, the semantic scalability scheme can be applied to various video categories.

Figura 4.2: Resumen y conclusiones del Documento 2.

Abstract Mobile devices need to provide more accurate and personalized information in a computing environment with a small screen and limited information retrieval functions. This paper presents a user-selectable recommendation system that reflects a user interest group by employing collaborative filtering as technique to provide useful information in a mobile environment. We form similar groups by simultaneously considering a user's information preferences and demographics. Then we reconstruct lists of a final recommendation based on what search results the similar demographic group has chosen. This is an optional filter for the search results. This means that we provide an interactive flexible recommendation list that considers a user's intent more actively, rather than unilaterally. We show the Mean Absolute Error result to evaluate the recommendation and finally show the realization of a prototype that is based on both the iPhone and Android phone environments.

...

5 Conclusion

Recommendation systems made significant progress over the last decade when numerous content-based, collaborative filtering, hybrid methods were proposed. However, despite all of these advances, the recommendation systems still require further improvements to make recommendation methods more effective [1]. So, in this paper, we described a flexible and interactive recommendation system in terms of Web 2.0 using collaborative filtering. Users can select similar groups who are more interesting and more important to them whenever they want. There are 15 cases that a user can select based on age, gender, location, and occupation. Finally, we implemented the prototype by the cases in the iPhone and android phone.

This paper finds it to be significant that in a mobile environment a user can select and change the interesting group easily to affect their own recommendation. Second, depending on a user's choice, minimizing the comparison group by eliminating unnecessary user groups can improve the efficiency of the system. Third, considering the location as an element to reflect the recommendation, this proposed system has scalability to ready a future ubiquitous environment.

Despite these significant points, this paper has limitations. We had difficulty to confirm the accuracy of location-based recommendations because of the limitation of a modified dataset. Also, we can't adopt the result for Korean users immediately, since the dataset is made based on US users. Even so, we hope that the issue presented in this paper will advance the research in the recommendation systems and our future research includes the next generation of recommendation systems using user's content consumption habits based in global culture.

Figura 4.3: Resumen y conclusiones del Documento 3.

Extracción de términos relevantes

En las Tablas 4.2, 4.3 y 4.4 se muestran los conjuntos de términos extraídos del resumen y las conclusiones de los documentos de prueba. Este proceso se realiza para todo el contenido de cada uno de los archivos procesados.

Obtención de relaciones

Para convertir los términos de las tablas 4.2, 4.3, 4.4 a una representación vectorial se aplica la Ecuacion 3.4. Un ejemplo de la matriz de salida se muestra en la Figura 4.4. En este ejemplo ilustrativo los pesos son elevados porque son muy pocos los términos, sólo se trabaja con tres documentos y sólo hay una coincidencia de un verbo con un sustantivo.

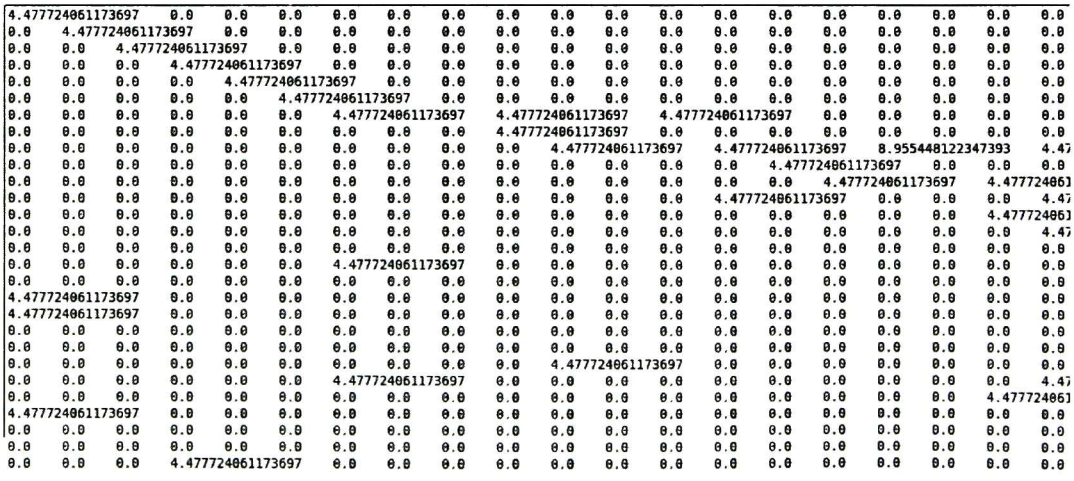


Figura 4.4: Fragmento de la matriz de salida para el cálculo de *tf-idf* del Documento 2.

sustantivos	verbos
design	exploit
treasure	make
kit	be
trend	propose
content sharing	increase
variability	support
role design	create
platform	reveal
game	explore
user	enable
contents	set
end user	recruit
evaluation result	reorganize
object	validate
dynamic	introduce
environment redeployment	generate
action	attain
subject	share
concept	provide
element	represent
gaming experience	play
user experience	configure
experience	allow
mechanisms	enhance
effort	
conclusion treasure	
environment	
trial game	
effectiveness	
context	
player	
user cooperation	

Tabla 4.2: Documento 1, lista de verbos y sustantivos

sustantivos	**verbos**
quality	decrease
home	annotate
frame	project
clip	shoot
purpose	be
evaluator	identify
scalability scheme	propose
level	reveal
conclusion	reduce
evaluation	reintegrate
bitstream	analyze
amount	achieve
bitrate	watch
bitstream size	attention
contents	extract
video content	apply
method	use
scalability	maintain
video	compress
visual	determine
information	test
plane	provide
tennis video	extend
camera motion	replace
background scene	compromise
watching video	show
option	detect
scheme	discard
dimension	
concept	
play clip	
result	
shot category	
content	

Tabla 4.3: Documento 2, lista de verbos y sustantivos

sustantivos	verbos
research	include
generation	affect
recommendation	form
group	present
user	minimize
paper	consider
comparison group	have
location	make
difficulty	choose
system	be
limitation	employ
recommendation system	confirm
progress	implement
dataset	propose
search result	require
case	change
collaborative	ready
accuracy	need
prototype	find
content	advance
method	improve
environment	reflect
mobile device	select
issue	evaluate
efficiency	content
interest group	use
error result	reconstruct
consumption habit	eliminate
list	adopt
user group	provide
result	show
recommendation list	
information	
realization	

Tabla 4.4: Documento 3, lista de verbos y sustantivos

A pesar de que los datos de ejemplo son pocos, en la Figura 4.5 se puede observar cómo se agrupan las relaciones verbo-sustantivo y en la Figura 4.6 se puede ver con más detalle la cantidad de relaciones verbo-sustantivo (clases) en las que se agrupa el Documento 2.

Para esta ejemplificación, el entrenamiento de la red SOM definió 11, 9 y 11 temas para los documentos 1, 2 y 3 con 22, 18 y 22 términos por tema respectivamente. Estas salidas son similares a las generadas cuando se procesan documentos completos.

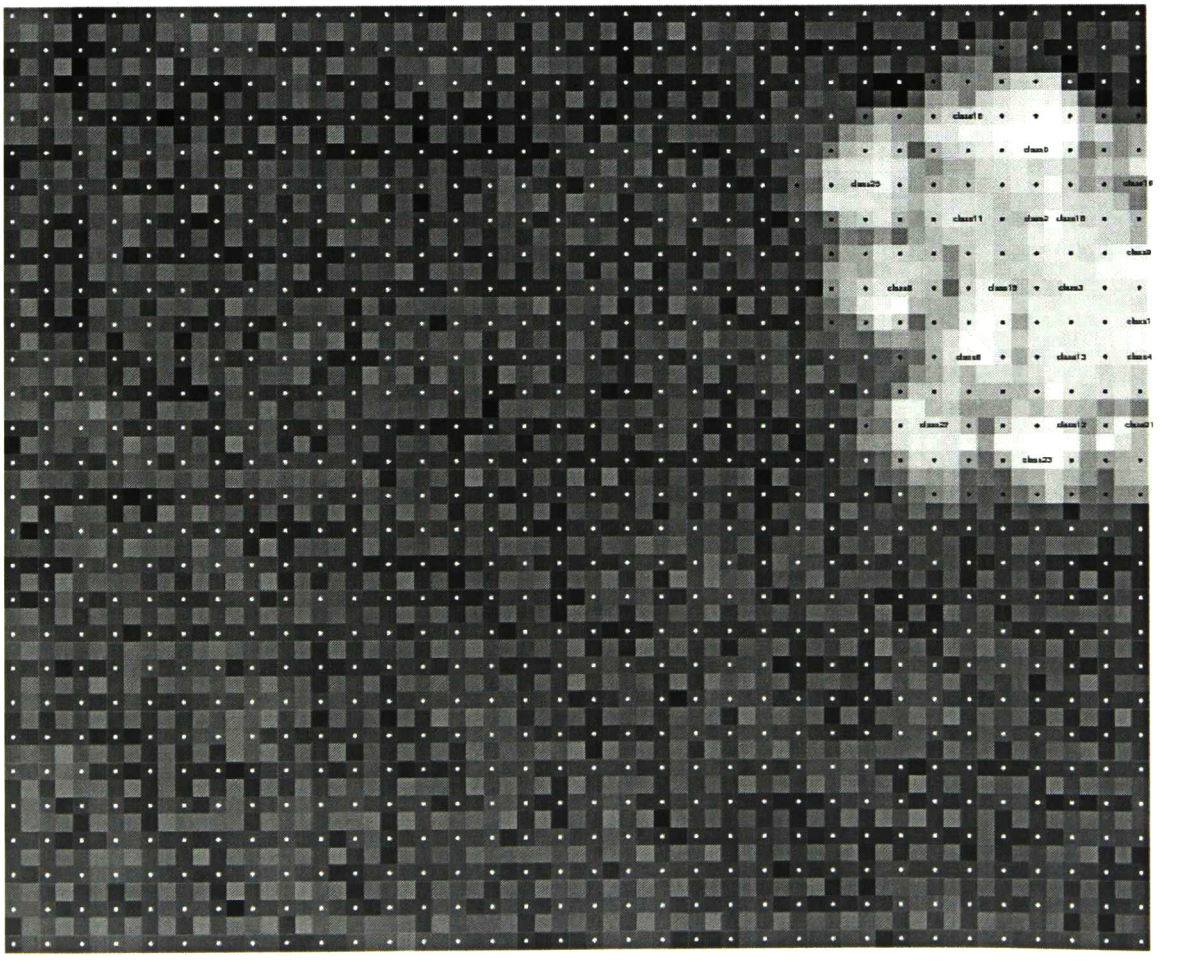


Figura 4.5: Salida de la red SOM para el Documento 2.

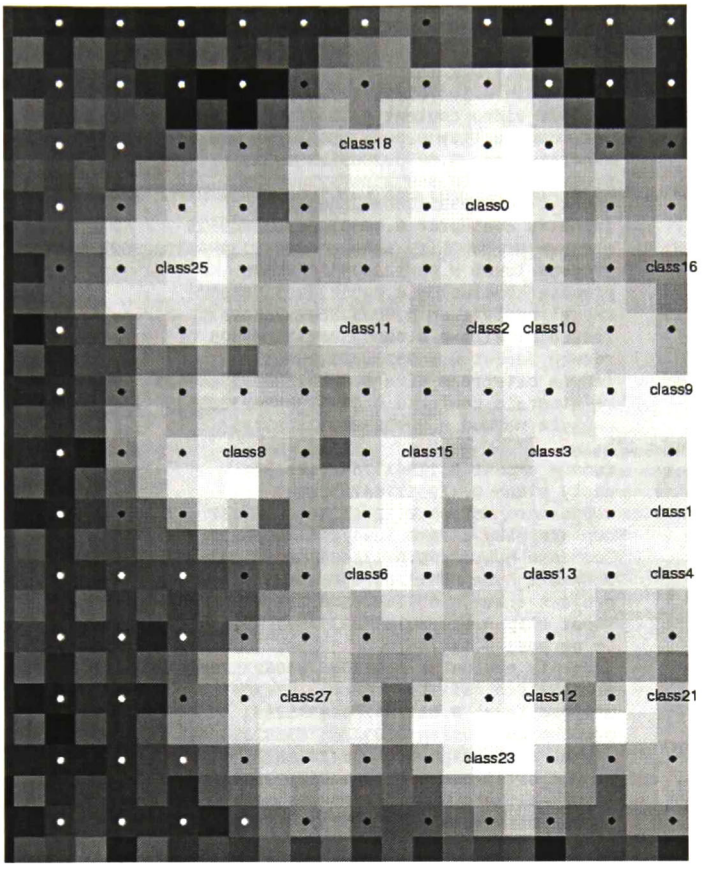


Figura 4.6: Clases que se generan como salida de la red SOM para el Documento 2.

La Figura 4.7 muestra un fragmento de la salida del algoritmo LDA durante la identificación de temas y relaciones por tema. En ella se listan dos de los nueve temas que corresponden al Documento 2. Cada tema tiene 18 relaciones de tipo **verbo-sustantivo/verbo-sustantivo compuesto**, las cifras que acompañan a cada relación representan la distribución de palabra-tema, por ejemplo $p(word_w|topic_t)$. Se puede observar que aunque una relación pueda aparecer en más de un tema, tendrá una distribución diferente en cada uno.

```
Topic 0th:
        reduce_bitrate 0.2
        show_result 0.2
        be_purpose 0.10476190476190476
        analyze_video_content 0.10476190476190476
        decrease_quality 0.009523809523809525
        annotate_home 0.009523809523809525
        project_frame 0.009523809523809525
        shoot_clip 0.009523809523809525
        identify_evaluator 0.009523809523809525
        propose_scalability_scheme 0.009523809523809525
        propose_level 0.009523809523809525
        propose_conclusion 0.009523809523809525
        reveal_evaluation 0.009523809523809525
        reduce_bitstream 0.009523809523809525
        reduce_amount 0.009523809523809525
        reduce_bitstream_size 0.009523809523809525
        reintegrate_contents 0.009523809523809525
        analyze_method 0.009523809523809525
Topic 1th:
        reduce_amount 0.12941176470588237
        apply_plane 0.12941176470588237
        use_camera_motion 0.12941176470588237
        replace_play_clip 0.12941176470588237
        decrease_quality 0.011764705882352941
        annotate_home 0.011764705882352941
        project_frame 0.011764705882352941
        shoot_clip 0.011764705882352941
        be_purpose 0.011764705882352941
        identify_evaluator 0.011764705882352941
        propose_scalability_scheme 0.011764705882352941
        propose_level 0.011764705882352941
        propose_conclusion 0.011764705882352941
        reveal_evaluation 0.011764705882352941
        reduce_bitstream 0.011764705882352941
        reduce_bitrate 0.011764705882352941
        reduce_bitstream_size 0.011764705882352941
        reintegrate_contents 0.011764705882352941
```

Figura 4.7: Agrupación de relaciones por tema generados por LDA para el Documento 2.

Representación gráfica

Al obtener los términos derivados para cada palabra relevante se debe asignar un identificador único a cada término para poder construir el grafo y establecer un peso, como se mostró en la sección 3.3.1. En la Figura 4.8 se puede ver un pequeño fragmento de los términos que se generan para el Documento 2. En la primera columna aparece el identificador asignado por WordNet a cada término relevante, adicionalmente se pueden asignar claves propias con valores negativos si el término no existe en WordNet o comparte su clave con una palabra de igual significado (ejemplo: immortalise, immortalize). En la segunda columna aparece el término que se encuentra originalmente

en el documento, la tercera y cuarta columnas son la clave y el término derivado[2]. En la quinta columna se establece a qué documento corresponden los términos y en la última columna el peso del término derivado, este valor varía dependiendo de si es un sinónimo, hiperónimo o hipónimo con un valor de 0.90, 0.75 y 0.60 respectivamente. En el caso del primer término, éste siempre valdrá 1.0. Cada una de estos renglones representa un enlace en el grafo que se va generando para enlazar los términos de cada documento y los documentos entre sí.

```
82143283        reveal          81065630        babble          Documento2.pdf  0.9
82143283        reveal          82533109        screen          Documento2.pdf  0.9
82143283        reveal          82631856        exhibit         Documento2.pdf  0.9
82143283        reveal          82631856        peach           Documento2.pdf  0.9
82143283        reveal          8952524         tell            Documento2.pdf  0.75
95210116        scalability     914530659       ratability      Documento2.pdf  0.9
95210116        scalability     95209822        quantifiability Documento2.pdf  0.75
95210116        scalability     95209822        measurability   Documento2.pdf  0.75
82327200        provide         81181559        power           Documento2.pdf  0.9
82327200        provide         82337699        bottom          Documento2.pdf  0.9
82327200        provide         82357561        toggle          Documento2.pdf  0.9
```

Figura 4.8: Fragmento de términos derivados de la consulta a WordNet para los documentos 1, 2 y 3.

La Figura 4.9 muestra cómo se empieza a generar paulatinamente el grafo a partir de la información de la Figura 4.8. En ésta se puede observar cómo es que se enlazan los términos. El grafo final para únicamente los tres documentos se ilustra en la Figura 4.10. Como se puede observar, la mayoría de los nodos no son visibles, sin embargo, es posible observar los nodos aislados y la enorme cantidad de términos y conexiones que existen. Si comparamos la información de las tablas 4.2, 4.3 y 4.4 con esta figura, se puede ver que se tiene mayor probabilidad de encontrar un documento a través de los sinónimos, hiperónimos e hipónimos, que a partir de los términos que contiene de manera sintáctica. Sería más complicado relacionar un documento si únicamente se consideran los términos de las tablas.

[2] *término derivado*: es un sinónimo, hiperónimo o hipónimo de una palabra relevente extraída de un documento.

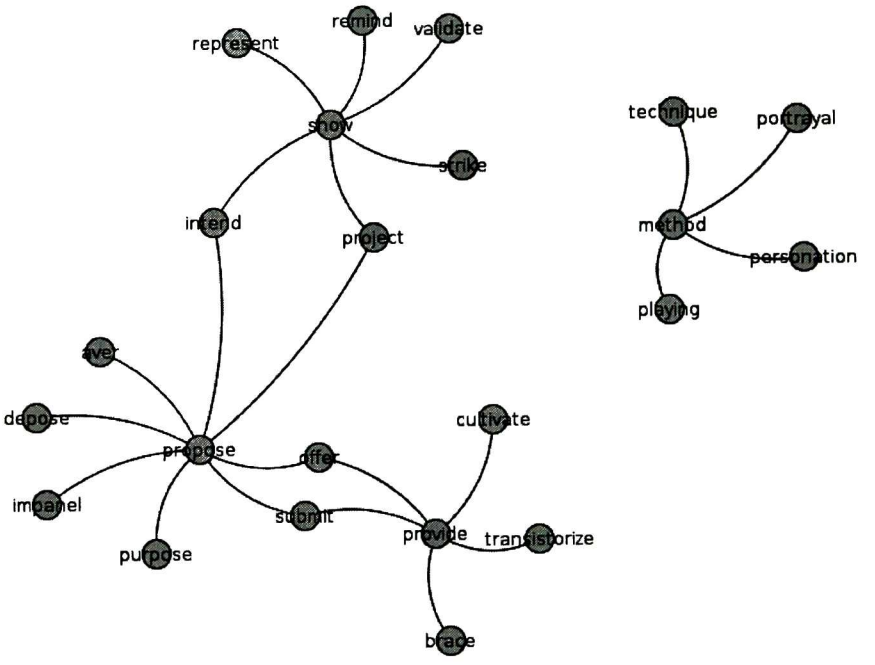


Figura 4.9: Ejemplo de la derivación de términos obtenidos de la consulta a WordNet para 4 palabras del Documento 2.

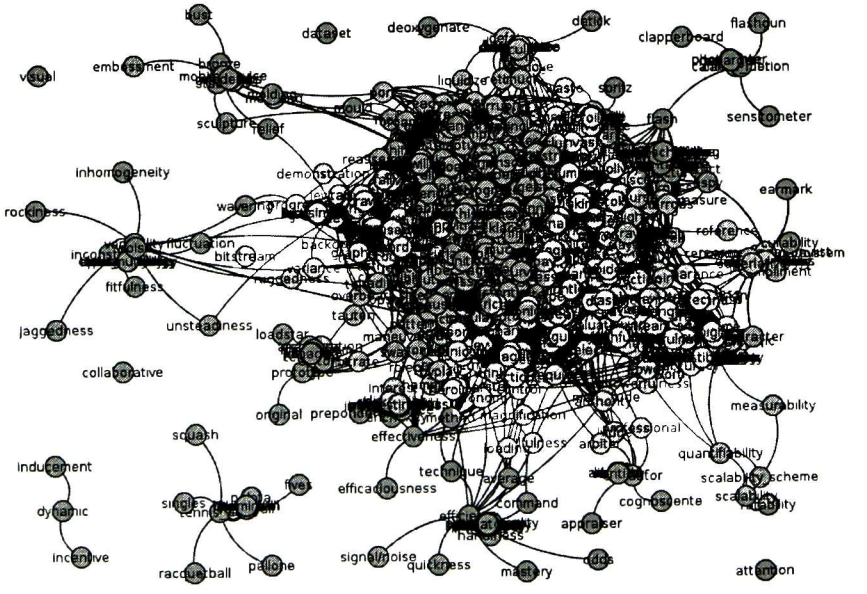


Figura 4.10: Grafo resultante de los términos derivados de los documentos 1, 2 y 3.

4.4 Evaluación de la metodología

La metodología propuesta como sistema de recuperación de información se comparó con las herramientas *Google Desktop*[3] y *SemanticVectors*[4]. Se eligió *Google Desktop* pues es una de las más usadas para la búsqueda de documentos en las computadoras de los usuarios. Por su parte *SemanticVectors* es una herramienta de búsqueda semántica desarrollada en Java con el modelo de vectores en donde las palabras y conceptos se representan como puntos en un plano [65]. Ambas aplicaciones operan sólo con los archivos del disco duro del usuario y realizan un proceso de indexación previo con los archivos que se le den como entrada, al igual que en esta propuesta este proceso se hace sólo un vez y consume tiempo.

Para la evaluación del cálculo local y cálculo global, cuya finalidad es medir la similaridad de documentos y recuperar información, se realizaron 31 consultas (de acuerdo al Teorema del Límite Central [21]). Los resultados de estas pruebas han sido medidos con las métricas *precision* y *recall* definidas en el capítulo 2.

La Tabla 4.5 muestra una lista de las consultas realizadas sobre el corpus de Springer. Las Tablas 4.6, 4.7 y 4.8 muestran las 31 consultas realizadas sobre el corpus 20 Newsgroups.

[3]Disponible en `http://googledesktop.blogspot.com/`, accesado en octubre del 2011
[4]Disponible en `http://code.google.com/p/semanticvectors/`, accesado en octubre del 2011

1	algorithmica
2	knowledge and information systems
3	multimedia tools and applications
4	software quality
5	supercomputing
6	minimizing the symmetric difference
7	the problem of similarity search on high dimensional data
8	image processing
9	substitution of improvisation with the common rules that can ensure acceptable quality levels
10	reducing the total power consumption in a significant amount
11	vertex does not disconnect the graph
12	build action rules from existing classification rules
13	multimedia content delivery to all mobile phones on all networks
14	software engineering
15	development of efficient multicore versions of our algorithms
16	strategy sets of simple paths
17	technique for evaluation of a data set
18	the process of embedding information into audio termed as audio watermarking
19	user friendly interface
20	a comparative study of parallel computing methods in terms of efficiency and effectiveness
21	a problem of partitioning a given finite set
22	the problem of similarity search on high dimensional data
23	a user interface for searchers, a retrieval object, and a searcher information
24	metric to measure the efficiency of a system
25	the optimum performance from the distributed computing system
26	issue for simplification of the boundaries of a region
27	framework for designing a fuzzy rule-based classifier.
28	method that recovers files
29	configuration Management
30	model for the evaluation of the estimated runtime of the different policies filler
31	representing planar graphs

Tabla 4.5: Consultas realizadas sobre el corpus de Springer

1	GUI toolkits
2	standard port addresses
3	The technician asked me about the drivers
4	computer store
5	Could the boards be incompatible with future releases of the OS?
6	repairs to a high-resolution monitor
7	the best audio device
8	change the paper type
9	how to extend the lifespan of a laptop battery
10	receive information on new equipment
11	unix commands

Tabla 4.6: Consultas realizadas sobre el corpus 20 Newsgroups para los temas `comp.os.ms-windows.misc` y `comp.sys.ibm.pc.hardware`

12	informations about digital controlled devices.
13	List of potential weaknesses in cryptographic codes
14	declassify the media by the overwrite method
15	modify the program code before running it
16	Laboratory tests for computer viruses
17	rules for deciphering codes
18	how to clean files that have been infected by viruses?
19	difference between encryption and encoding algorithms
20	strong public key encryption and digital signature application using the RSA algorithm
21	What rules are necessary to protect the privacy of electronic documents?

Tabla 4.7: Consultas realizadas sobre el corpus 20 Newsgroups para el tema `sci.crypt`

22	views on the conflict in Israel
23	international Congresses
24	discrimination based on race, color or sex
25	energy consumption
26	American Civil War
27	someone can be charged several times in a single crime?
28	predictions about the unemployment rate
29	Statistics on drug use in adolescents
30	Do you have found new sources of oil?
31	President Clinton's scandals

Tabla 4.8: Consultas realizadas sobre el corpus 20 Newsgroups para el tema `talk.religion.misc`

4.5 Resultados de los experimentos

Analizando las consultas realizadas sobre el corpus de Springer de la Tabla 4.5, en la Figura 4.11 se puede observar que *Google Desktop* en 10 de los 31 casos no logra recuperar información relevante a pesar que las frases están relacionadas con los temas. Esto puede deberse a las diversas formas de expresión de una misma palabra, expresiones que los buscadores basados en palabras clave no pueden asociar. Caso contrario a la aplicación desarrollada, que se comporta de manera más estable que *Google Desktop* (debido a que sus valores de *precision* van de 0.15 al 0.7, es decir nunca caen en cero y tampoco recupera una enorme cantidad de documentos). En la gráfica de la Figura 4.11 se pueden ver algunos los casos en que *Google Desktop* parece tener un mejor rendimiento, pero esto se debe a que la cantidad de documentos recuperados es muy pequeña, como se puede observar en la Figura 4.12 para las consultas 6, 11, 23, 27 y 28 del corpus de Springer.

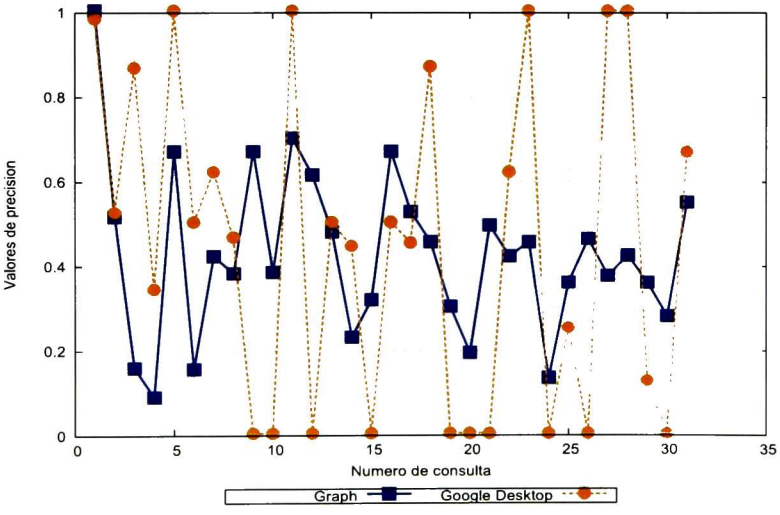


Figura 4.11: Gráfica comparativa de los valores de *precision* del Grafo obtenido y *Google Desktop* para el corpus de Springer

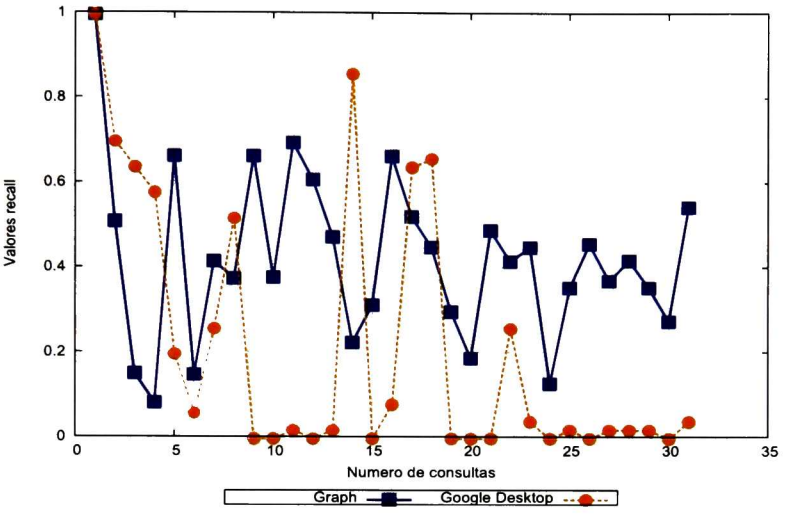


Figura 4.12: Gráfica comparativa de los valores de *recall* del Grafo obtenido y *Google Desktop* para el corpus de Springer

También se puede observar que la aplicación desarrollada en todas las consultas es capaz de recuperar información. Esto se debe a la relación que existe entre los temas de los documentos ganadores con el resto del corpus.

En las gráficas de las Figuras 4.13 y 4.14 se muestran los resultados de *precision* y *recall* para la aplicación *SemanticVectors* de las consultas de la Tabla 4.5 hechas sobre el corpus de Springer . Para hacer una evaluación justa con respecto a nuestra aplicación, se ha realizado una pequeña modificación al código de esta aplicación. Inicialmente solicita al usuario la cantidad de documentos desea recuperar, pero internamente este software fija un umbral, de este modo si la cantidad de documentos recuperados supera el umbral, se agregan más documentos al resultado final y se eliminan en caso de que el umbral supere la cantidad deseada por el usuario. Esta condición se ha eliminado de la aplicación para que retorne sólo los resultados que estén dentro del umbral establecido y así evaluar de forma adecuada con las métricas establecidas en el capítulo 2.

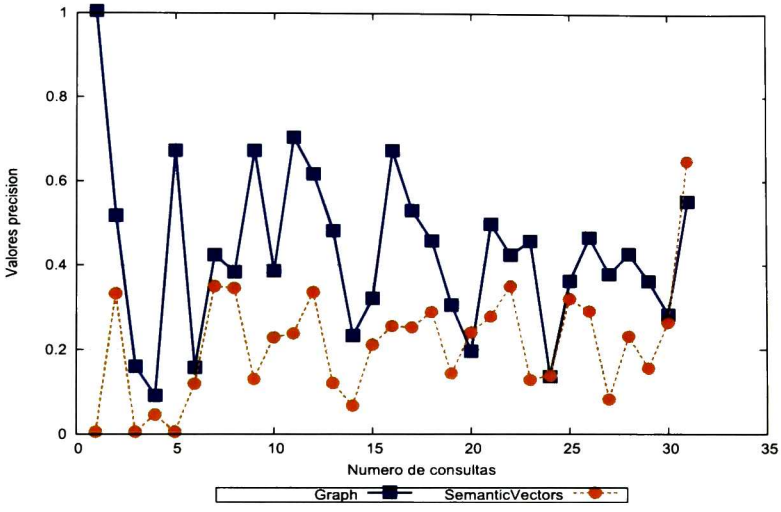


Figura 4.13: Gráfica comparativa de los valores de *precision* del Grafo obtenido y *SemanticVectors* para el corpus de Springer

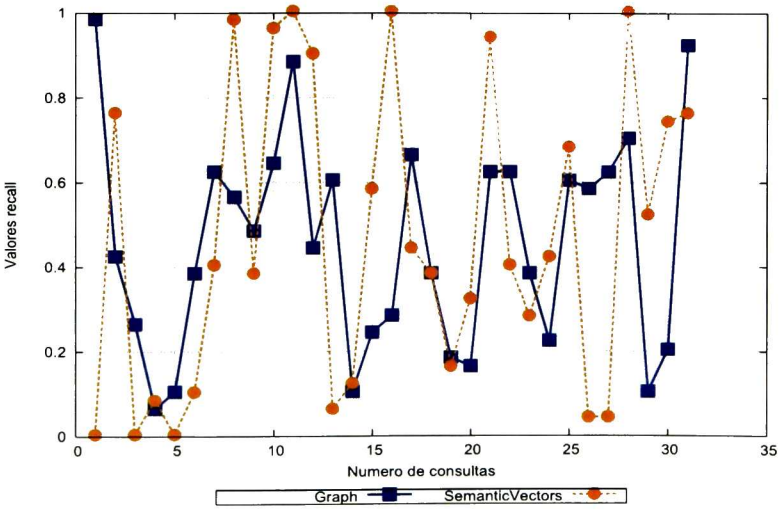


Figura 4.14: Gráfica comparativa de los valores de *recall* del Grafo obtenido y *SemanticVectors* para el corpus de Springer

		Propuesta	Google Desktop	Propuesta	Google Desktop
Consulta	**Etiqueta**	**Precision**	**Precision**	**Recall**	**Recall**
1	Algorithmica	1	0.9803921569	0.98	1
2	Knowledge and Information Systems	0.512195122	0.5223880597	0.42	0.7
3	Multimedia Tools and Applications	0.1547619048	0.8648648649	0.26	0.64
4	Software Quality	0.0857142857	0.3411764706	0.06	0.58
5	Supercomputing	0.6666666667	1	0.1	0.2
6	Algorithmica	0.152	0.5	0.38	0.06
7	Knowledge and Information Systems	0.4189189189	0.619047619	0.62	0.26
8	Multimedia Tools and Applications	0.3783783784	0.4642857143	0.56	0.52
9	Software Quality	0.6666666667	0	0.48	0
10	Supercomputing	0.380952381	0	0.64	0
11	Algorithmica	0.6984126984	1	0.88	0.02
12	Knowledge and Information Systems	0.6111111111	0	0.44	0
13	Multimedia Tools and Applications	0.4761904762	0.5	0.6	0.02
14	Software Quality	0.2272727273	0.4432989691	0.1	0.86
15	Supercomputing	0.3157894737	0	0.24	0
16	Algorithmica	0.6666666667	0.5	0.28	0.08
17	Knowledge and Information Systems	0.5238095238	0.4507042254	0.66	0.64
18	Multimedia Tools and Applications	0.4523809524	0.8684210526	0.38	0.66
19	Software Quality	0.3	0	0.18	0
20	Supercomputing	0.1904761905	0	0.16	0
21	Algorithmica	0.4920634921	0	0.62	0
22	Knowledge and Information Systems	0.4189189189	0.619047619	0.62	0.26
23	Multimedia Tools and Applications	0.4523809524	1	0.38	0.04
24	Software Quality	0.130952381	0	0.22	0
25	Supercomputing	0.3571428571	0.25	0.6	0.02
26	Algorithmica	0.4603174603	0	0.58	0
27	Knowledge and Information Systems	0.3734939759	1	0.62	0.02
28	Multimedia Tools and Applications	0.421686747	1	0.7	0.02
29	Software Quality	0.3571428571	0.125	0.1	0.02
30	Supercomputing	0.2777777778	0	0.2	0
31	Algorithmica	0.5476190476	0.6666666667	0.92	0.04

Tabla 4.9: Tabla comparativa de los valores de *precision* y *recall* de la aplicación desarrollada y *Google Desktop* sobre el corpus de Springer

Desafortunadamente *SemanticVectors* ha dado muy malos resultados. En la gráfica de la Figura 4.13 puede verse cómo esta aplicación no logra superar en cuanto a precisión a nuestra propuesta. Para la cantidad de documentos recuperados de la gráfica de la Figura 4.14 ocurre un caso opuesto al de *Google Desktop*, es decir los resultados se elevan sólo porque retorna una gran cantidad de documentos (de 144 a 215 cuando el corpus es igual a 250). Si la aplicación retorna todos o la mayoría de los documentos entre todos ellos tendrán que encontrarse los 50 que corresponden al grupo adecuado.

Analicemos por ejemplo el caso de la primera consulta ("algorithmica"), esta palabra se encuentra escrita en 50 archivos, pero como el método se basa en representación vectorial, si esta palabra está aislada no se podrá formar un vector para dicha consulta, por lo tanto no logra recuperar ningún documento que se relacione con el término de entrada. Ocurre lo mismo con el resto de las consultas para las cuales retorna poca o nula información, ya que si la consulta tiene varias palabras pero estas no aparecen juntas ocurrirá el mismo caso, es decir, no existirá un vector que coincida con la consulta.

La ventaja que tiene la propuesta presentada sobre *SemanticVectors* es la conexión entre documentos ya que, a pesar de encontrar algunos términos dentro de los documentos esto no significa que sea justo la información que se está buscando.

En las gráficas de las Figuras 4.15, 4.17 y 4.19 se muestra cómo para el corpus *20 Newsgroups*, una aplicación de búsqueda convencional que se basa en palabras clave como lo es *Google Desktop*, no logra recuperar información a pesar de que las consultas no son complejas y el grupo del cual se puede recuperar información es mayor. También se puede ver cómo pareciera que la aplicación basada en vectores (*SemanticVectors*) tiene una *precision* un poco más alta que la propuesta presentada, si analizamos las gráficas de las Figuras 4.16, 4.18 y 4.20 de los valores de las Tablas 4.10 y 4.11 veremos como sus valores de recall son notablemente mucho más elevados a comparación de las otras 2 aplicaciones, la razon es que esta aplicación retorna en promedio 1888 documentos por

		Propuesta	Google Desktop	SemanticVectors
Consulta	*Etiqueta	Precision	Precision	Precision
1	1	1.000000000	1.000000000	0.643630308
2	1	0.457800512	0.000000000	0.562947368
3	1	0.571428571	0.000000000	0.541728763
4	1	1.000000000	0.937500000	0.555555556
5	1	0.437500000	0.000000000	0.546951748
6	1	0.636942675	0.000000000	0.723577236
7	1	0.595744681	0.000000000	0.611940299
8	1	0.319727891	0.000000000	0.626633166
9	1	0.369565217	0.000000000	0.451693852
10	1	0.410774411	0.000000000	0.402542373
11	1	0.456549935	0.500000000	0.525950513
12	2	0.405228758	0.000000000	0.367857143
13	2	0.846153846	0.000000000	0.650176678
14	2	1.000000000	1.000000000	0.345060893
15	2	0.600000000	1.000000000	0.296460177
16	2	0.500000000	0.000000000	0.300943009
17	2	0.714285714	0.000000000	0.364025696
18	2	0.700000000	0.000000000	0.313074205
19	2	1.000000000	0.000000000	0.505681818
20	2	0.600000000	1.000000000	0.522157996
21	2	0.444444444	0.000000000	0.542772861
22	3	1.000000000	0.000000000	0.476190476
23	3	0.634831461	0.000000000	0.278260870
24	3	0.487804878	0.000000000	0.792452830
25	3	0.439024390	0.000000000	0.143236074
26	3	1.000000000	0.333333333	0.360435876
27	3	1.000000000	0.000000000	0.149396378
28	3	1.000000000	0.000000000	0.256013746
29	3	1.000000000	1.000000000	0.166409861
30	3	0.586206897	0.000000000	0.180274859
31	3	1.000000000	0.000000000	0.6516129032

* Etiqueta 1= comp.os.ms-windows.misc y comp.sys.ibm.pc.hardware, Etiqueta 2 = sci.crypt y Etiqueta 3 = talk.religion.misc

Tabla 4.10: Tabla comparativa de los valores de *precision* de la aplicación desarrollada, *Google Desktop* y *SemanticVectors* para el corpus de *20 Newsgroups*

consulta. Es el punto que mencionábamos anteriormente, si retorna una cantidad muy grande de documentos estadísticamente tendrá más posibilidades de que entre los documentos se encuentren los que pertenecen a la clase adecuada.

Un punto importante en las aplicaciones de recuperación de información es no retornar demasiados resultados ya que para un usuario es inútil obtener como respuesta 1888 documentos puesto que no podría analizarlos todos, en este caso sería más eficiente abrir un archivo al azar. La misma tabla nos muestra cómo la propuesta presentada retorna mucho menos documentos por consulta. El objetivo aquí es encontrar un equilibrio entre ambas métricas (*precision* y *recall*), para mostrar resultados más precisos entre una cantidad razonable de archivos devueltos.

		Propuesta	Google Desktop	SemanticVectors
Consulta	*Etiqueta	Precision	Precision	Precision
1	1	0.003083248	0.000513875	0.397225077
2	1	0.183967112	0.000000000	0.687050360
3	1	0.002055498	0.000000000	0.373586845
4	1	0.000513875	0.007708119	0.662898253
5	1	0.010791367	0.000000000	0.972764645
6	1	0.102774923	0.000000000	0.274409044
7	1	0.014388489	0.000000000	0.294964029
8	1	0.024152107	0.000000000	0.640801644
9	1	0.034943474	0.000000000	0.369989723
10	1	0.062692703	0.000000000	0.244090442
11	1	0.542651593	0.000513875	0.895683453
12	2	0.062563068	0.000000000	0.415741675
13	2	0.011099899	0.000000000	0.371342079
14	2	0.002018163	0.001009082	0.257315843
15	2	0.012108981	0.001009082	0.811301715
16	2	0.002018163	0.000000000	0.740665994
17	2	0.005045409	0.000000000	0.171543895
18	2	0.007063572	0.000000000	0.447023209
19	2	0.002018163	0.000000000	0.718466196
20	2	0.003027245	0.001009082	0.820383451
21	2	0.004036327	0.000000000	0.557013118
22	3	0.002580645	0.000000000	0.309677419
23	3	0.145806452	0.000000000	0.247741935
24	3	0.025806452	0.000000000	0.054193548
25	3	0.023225806	0.000000000	0.139354839
26	3	0.002580645	0.001290323	0.554838710
27	3	0.002580645	0.000000000	0.383225806
28	3	0.001290323	0.000000000	0.192258065
29	3	0.002580645	0.002580645	0.278709677
30	3	0.021935484	0.000000000	0.287741935
31	3	0.002580645	0.000000000	0.651612903

* Etiqueta 1= comp.os.ms-windows.misc y comp.sys.ibm.pc.hardware, Etiqueta 2 = sci.crypt y Etiqueta 3 = talk.religion.misc

Tabla 4.11: Tabla comparativa de los valores de *recall* para la aplicación desarrollada, *Google Desktop* y *SemanticVectors* para el corpus de *20 Newsgroups*

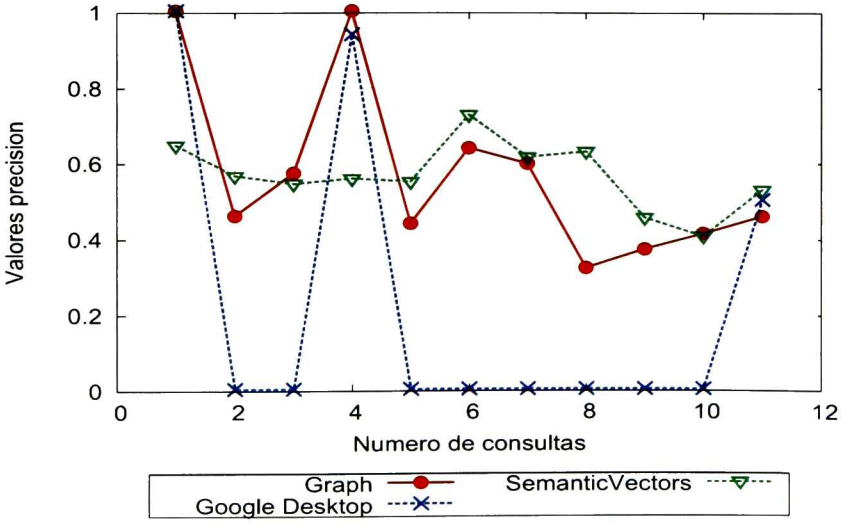


Figura 4.15: Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.6

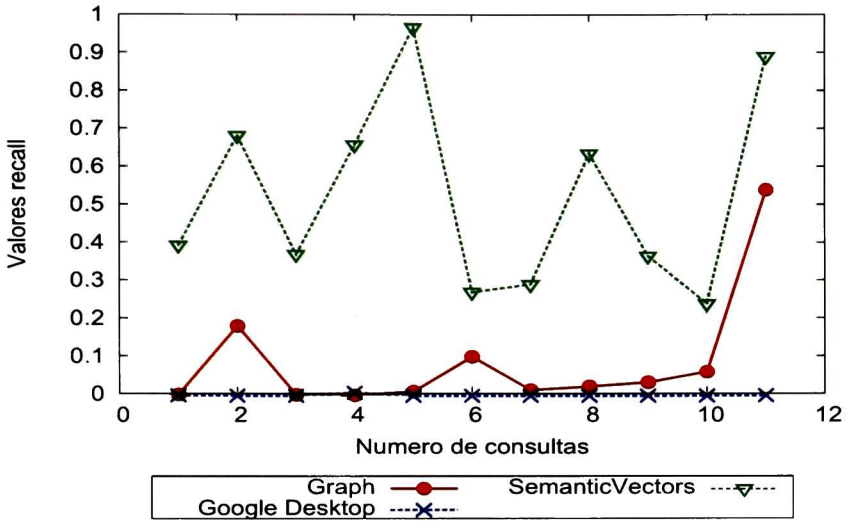


Figura 4.16: Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.6

Para las gráficas 4.17 y 4.18 en el que los temas tienen menos intersección se aprecia que el objetivo planteado para el uso de este corpus se ha logrado, es decir que la aplicación logre mejores resultados con grupos de datos visiblemente menos relacionados. Para estos dos temas la aplicación *SemanticVectors* retorna un 28.8 % y 44.3 % (del 100 % que suman todos los documentos) más documentos en promedio de los que corresponden a la clase. En lo que respecta a *Google Desktop*, su rendimiento sigue siendo más bajo que de la herramientas semánticas.

Analicemos por ejemplo el caso de la consulta 10 de la Tabla 4.8. En la consulta 10 las palabra *president* aparece 338 veces y el nombre *Clinton* aparece 392. Sin embargo *Google Desktop* no retorna ningún archivo en la consulta. Esto se debe a que a pesar de que estos 2 términos pudieran aparecer o no juntos en diferentes archivos, si la tercera palabra no aparece acompañando a una de la 2 anteriores, entonces no se encontrará un archivo que coincida con la consulta de entrada, a menos que la consulta se modifique, es decir que se busque cada palabra por separado. Esto no significa que los 3 términos no estén juntos en una oración o documento, es más bien que el problema con este tipo de aplicaciones no considera oraciones o ideas completas. Para este caso específico los tres

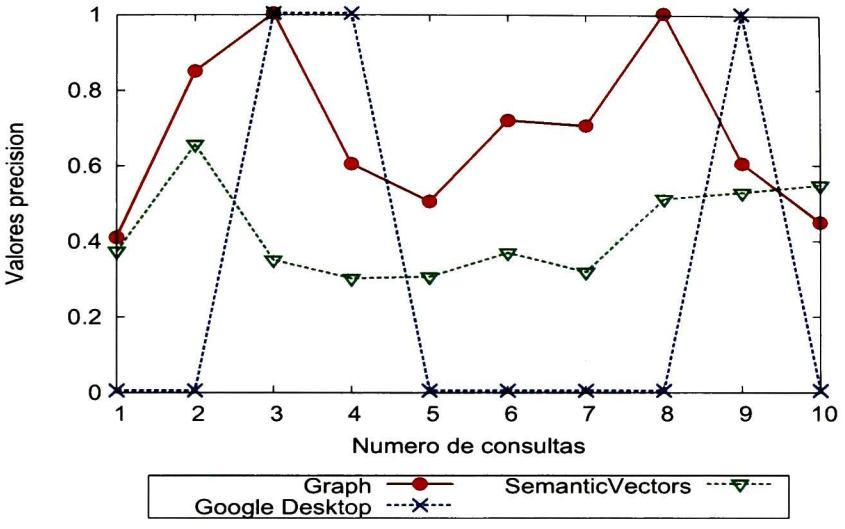


Figura 4.17: Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.7

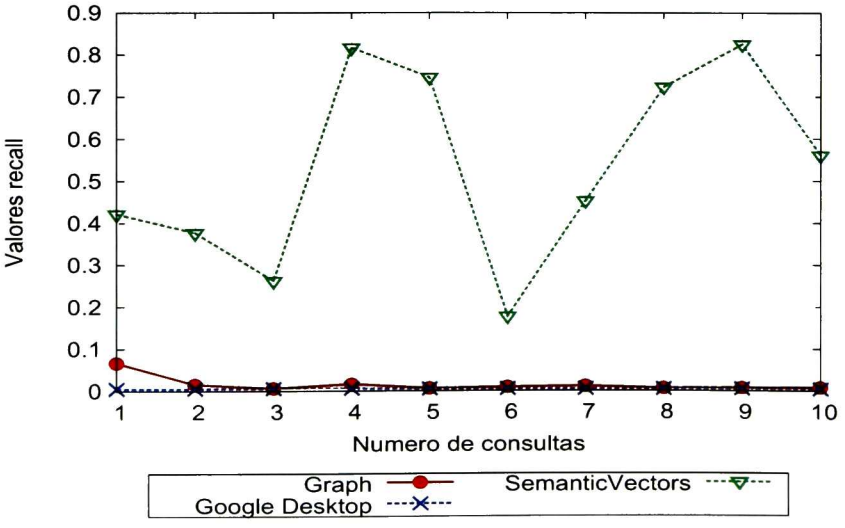


Figura 4.18: Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.7

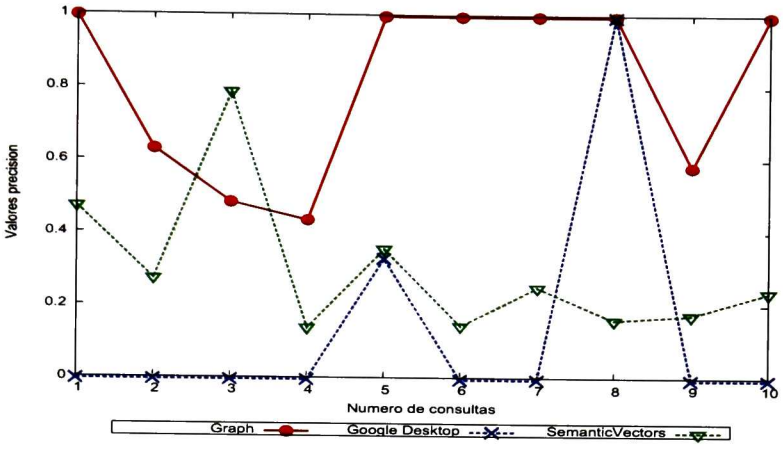


Figura 4.19: Gráfica comparativa de los valores de *precision* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.8

términos se encuentran separados por uno o más términos dentro de una oración.

Los tiempos de ejecución de la aplicación desarrollada pueden evaluarse de acuerdo a cada una de las etapas de la metodología. Para ello se han tomado en cuenta tiempos promedios para cada corpus y para cada etapa. En la Tabla 4.12 se muestra una comparativa de estos tiempos. Una consideración importante es que estos valores corresponden al tiempo de preprocesamiento, no a las consultas pues estas toman muy poco tiempo, aproximadamente 43 segundos.

El tiempo de indexado para *Google Desktop* es de 1 hora, en el caso *SemanticVectors* el proceso de indexado se hace en 2 fases, la primera toma 9.10s y para la segunda 7.5s aproximadamente.

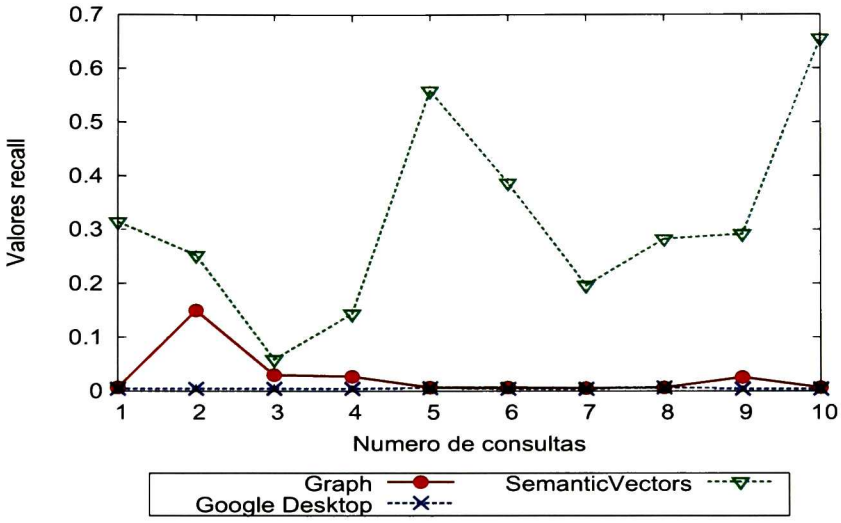


Figura 4.20: Gráfica comparativa de los valores de *recall* del Grafo obtenido, *Google Desktop* y *SemanticVectors* para el corpus *20 Newsgroups* para las consultas de la Tabla 4.8

Etapa	**Springer**	**20 Newsgroups**
Extracción de texto	0.018 hrs	0.0381 hrs
Extracción de términos relevantes	36 hrs	63.40 hrs
-Representación vectorial	0.145 hrs	0.021 hrs
-SOM	4.56 hrs	0.096 hrs
-LDA	0.0036 hrs	0.0033 hrs
Recuperación de significados	5.04 hrs	18.17 hrs
Construcción del grafo	76.336 hrs	54.7 hrs

Tabla 4.12: Tiempos de ejecución por cada etapa en la construcción del grafo

4.6 Resumen

En este capítulo se han presentado los resultados de una aplicación basada en la búsqueda a través de un grafo. El principal objetivo de la aplicación ha sido encontrar un balance entre la cantidad de información que se retorna al usuario y una alta precisión en las consultas sin limitarse a descartar información que no pertenezca estrictamente a un solo grupo de datos. En este sentido los dos primeros puntos son cubiertos por la ecuación denominada cálculo local.

Para cubrir el último punto relacionado a encontrar documentos cuyos temas se intercepten con otros documentos, se ha ideado el cálculo global, el cual se puede ver como un *cluster* dinámico de documentos en el que el grupo principal cambiará de acuerdo a las necesidades de cada solicitud de información. Los pesos que se le asignen a cada documento tanto con el cálculo local como el global en conjunto con la Ecuación 3.19 son los encargados de limitar el tamaño del grupo resultante.

Bajo estas condiciones, a lo largo del capítulo se han mostrado breves ejemplos del procesamiento y resultados requeridos en cada fase de la implementación. Para cada uno de los resultados mostrados en la etapas de preprocesamiento se pueden aislar características del grupo de datos que se procesa, como por ejemplo la cantidad de temas por documento y la separación entre los propios temas de cada documentos como se pueden ver en los resultados de los algoritmos LDA y SOM.

En este capítulo también se han presentado gráficas en las cuales se puede apreciar el rendimientos de nuestro enfoque en comparación con aplicaciones que utilizan métodos tradicionales como la búsqueda por palabras clave y el modelos de espacio vectorial.

4.7 Conclusiones

En este capítulo se han mostrado los resultados de la validación de la metodología con un conjunto de datos representativo del área de minería de datos con diferentes tipos de consultas. Se han tomando en consideración las referencias de la sección 1.1 propuestas por los creadores del buscador Hakia, y con las cuales se puede concluir que se han tratado de cubrir estas necesidades en su mayoría.

Como resultado de las validaciones presentadas en este capítulo y los experimentos realizados durante el desarrollo de esta tesis, concluimos que es posible utilizar un grafo como una forma de representación semántica de documentos. El proceso de construcción del grafo se logra de forma automática sin necesidad de realizar ningún cambio en la forma en la se que encuentran constituidos actualmente los diferentes tipos de archivos en los que se almacena diversidad de información. En otras palabras, podemos crear una capa de abstracción entre la información y el usuario.

Además de este aspecto, el prototipo de software construido a demostrado que los algoritmos de búsqueda propuestos son capaces de ubicar grupos de datos semánticamente conectados lo cual quedó demostrado la comparación que se ha hecho con las herramientas *Google Desktop* y *SemanticVectors*.

Observando los diagramas de conexión de términos como el de la Figura 4.10 podemos concluir que existirán grupos de datos altamente relacionados, este es uno de los grandes problemas a los que se enfrenta la búsqueda semántica, pero como pudimos constatar en los experimentos, el uso de las dos ponderaciones de similitud propuestas son eficientes al aislar estos grupos, tomando primero aquellos que se relacionan más con la consulta escrita por el usuario.

5

Conclusiones y trabajo futuro

En este capítulo se presentan un análisis basado en los resultados mostrados en los capítulos anteriores para presentar una breve discusión de la metodología, las ventajas y desventajas de la propuesta y algunos aspectos del trabajo futuro.

En este trabajo de tesis se abordó la problemática de la búsqueda y recuperación de información en el que, para llegar a las conclusiones que se citan en este capítulo, se hizo una investigación en la literatura existente, en el estado del arte, un análisis de las propuestas planteadas en esta área, una selección de grupos de datos para llevar a cabo dos casos de estudio y de ese modo verificar que esta propuesta ha logrado sus objetivos. Lo anterior logró la integración de los elementos necesarios para finalizar este proyecto.

Lo que se lista a continuación son algunas limitaciones y dificultades que se han tenido durante el desarrollo de esta investigación, así como las conclusiones a las que se llegaron con este proyecto.

5.1 Dificultades técnicas

Durante el desarrollo de esta tesis han sido diversas las dificultades técnicas que se ha presentado , entre las cuales cabe mencionar algunas que fueron particularmente esenciales para la favorable finalización de esta tesis.

- La dificultad de ubicar los diferentes problemas de los sistemas de búsqueda actuales para así poder presentar una mejora a lo existente en la literatura. En este trabajo no se pretende sugerir que esta sea una solución definitiva a todos esos problemas ya que hay aspectos de escapan al ámbito de esta tesis.

- La inversión de tiempo en estudiar, desarrollar y probar diferentes tipos de algoritmos que mejoran las características deseadas para construir la representación semántica, aunque no se hayan empleado finalmente.

- Las diversas pruebas que se realizaron para encontrar una forma adecuada para almacenar físicamente la estructura del grafo, problema del que se desglosan por ejemplo la asignación de claves por término, ponderaciones para los enlaces, problemas de almacenamiento, entre otros.

- Buena parte del tiempo que se ha invertido en la elaboración de esta tesis lo ha consumido el preprocesamiento de los datos, dado que en cada fase los algoritmos utilizados requieren cierto número de iteraciones para obtener probabilísticamente la información más adecuada.

- El problema para idear una ecuación de cálculo de similitud que hiciera un buen uso de la notación semántica propuesta.

5.2 Limitaciones

Aunque en el marco de trabajo se obtuvieron buenos resultados, existen varias limitaciones en la cobertura de este trabajo.

- Únicamente pueden procesarse documentos del idioma inglés. Afortunadamente en la actualidad el proyecto WordNet tiene algunos proyectos derivados de su base de datos que están disponibles en otros idiomas. Esto también puede tomarse a consideración como trabajo futuro.

- Este trabajo ha mostrado su desempeño frente a documentos reales, como artículos científicos, en los que se ha encontrado que la limpieza de los datos es un paso esencial. Pero no se realizaron pruebas para archivos ya preprocesados que utilizan otras técnicas, es decir no se pudo constatar el comportamiento del método frente a documentos que ya tienen una estructura semántica definida (como archivos OWL o RDF). Aunque el uso de éstos sea poco común seria un experimento bastante interesante.

- La aplicación de prueba se limita sólo a un prototipo dado que el refinamiento de una aplicación consumiría demasiado tiempo de desarrollo, tiempo que no se dispone para la finalización de esta tesis. Además de que aún existen algunos huecos en el área de computación para lograr un cambio verdaderamente significativo en el tiempo de ejecución, por ejemplo la paralelización de estructuras de datos globales como los grafos.

5.3 Contribuciones

En esta tesis se han abordado varios de los puntos relacionados con el problema de la Recuperación de Información. A continuación se enumeran las contribuciones hechas en este trabajo.

- Se ha obtenido un modelo de representación semántica de documentos en inglés. Esta metodología no sólo nos lleva a una representación visual de los datos, también permite extraer

las características de los mismos y sus dependencias semánticas. Con ello esta metodología puede utilizarse para otros procesos de la minería de datos, como por ejemplo el *clustering* de documentos.

- Se ha contribuido al estado del arte al desarrollar un método de organización semántica de documentos. El resultado final de la metodología propuesta en la sección 3.1 nos da como resultado una indexación completa de los datos que se toman como entrada, lo que facilita la búsqueda de información sobre dicha estructura.

- Se ha contribuido al estado del arte con un prototipo que permite validar los aspectos del modelo propuesto. En el capítulo 3 se dan los detalles de los algoritmos, parámetros y herramientas para el desarrollo de una aplicación que implementa la metodología y algoritmos propuestos.

- Se han validado los aspectos de esta propuesta mediante el desarrollo de una aplicación que ha sido probada teniendo en cuenta las métricas de evaluación de la recuperación de información: *precision* y *recall*. En el capítulo 4 se muestran los resultados de los experimentos con dos grupos de documentos utilizando la aplicación desarrollada en el capítulo 3. Se compararon los resultados de las métricas *precision* y *recall* de la aplicación basada en grafos contra una aplicación basada en palabras clave (*Google Desktop*) y una aplicación de búsqueda semántica (*SemanticVectors*).

5.4 Conclusiones

A continuación se describen las conclusiones a las que se han llegado con base a los puntos anteriores.

En este trabajo de tesis se ha descrito, evaluado y comprobado una forma para la representación de información de documentos tomando como base los mapas conceptuales para formar relaciones semánticas. Donde un mapa conceptual es una red de conceptos en la que los nodos son términos o palabras y los enlaces entre los nodos representan la relaciones semánticas. Como se planteó en el capítulo 3, el problema no se basa sólo en desarrollar un método de búsqueda, el objetivo es proponer una estructura semántica, como lo es el grafo, para imitar el proceso de construcción de conocimiento como lo realiza un ser humano.

En conjunto los algoritmos de búsqueda, similaridad y cantidad de documentos retornados planteados en este trabajo contribuyen a la solución al problema de la recuperación de información. Los algoritmos y fórmulas presentados en este trabajo conforman una interesante herramienta que facilita la búsqueda de información en los documentos.

Las desventajas principales de la implementación presentada están relacionadas con el tiempo de procesamiento requerido para indexar los datos. Sin embargo, es posible paralelizar estos procesos en la mayoría de sus fases ya que en cada una de ellas los archivos de entrada son analizados uno por uno. De este modo el proceso de paralelización consistiría en realizar el análisis y procesamiento de datos para más de un archivo de forma simultánea. Este proceso queda como trabajo futuro ya que el objetivo para este trabajo de tesis era probar la ventaja del método propuesto. Refinar una aplicación requería de un mayor tiempo del que no se dispone.

A pesar de las desventajas, la verificación de los algoritmos dio muy buenos resultados. Se realizaron varias pruebas con escenarios diferentes, uno de ellos ampliamente utilizado en el área de Recuperación de Información con una muestra representativa de cada tema para el conjunto de datos presentado, con lo cual se concluyó que el método propuesto supera a otros enfoques semánticos como los basados en vectores. Además de este análisis, se realizaron pruebas con conjuntos de datos reales, es decir con artículos académicos, para cumplir con el objetivo de un buscador de

información. Gracias al análisis de ambos grupos de pruebas en los que a pesar de que en algunas consultas los resultados no muestran una precisión del cien por ciento, se concluye que esto se debe a que los documentos que se retornan tienen intersección con más de una clase; lo que demuestra que las búsquedas se basan en las ideas contenidas en los documentos y no sólo en las palabras. Normalmente en una búsqueda real no se consulta un grupo de archivos bien clasificados sino que se intenta extraer información de diversos grupos de documentos sin tener conocimiento previo de la relación que hay entre todos los archivos. El que el enfoque propuesto logre encontrar esas relaciones es una característica destacable.

Otro tema notable en el enfoque propuesto es la cantidad de documentos retornados. Consideramos que la cantidad inicialmente definida por la aplicación es una buena opción para el usuario, pero creemos que también se debe dar libertad al usuario para poder definir la cantidad de documentos que desea que se retornen en una consulta, si en algún momento el usuario no está de acuerdo con la cantidad que se le presenta. Esto no significa que la medida propuesta para calcular la similitud de los documentos con la consulta tenga alguna deficiencia. Esta problemática representa un caso de estudio más amplio e independiente que queda fuera del alcance de este trabajo de tesis. Si bien se ha considerado un número dinámico de los documentos a retornar es porque partimos de la idea de crear un método que no requiera intervención del usuario.

Podemos concluir entonces que el enfoque propuesto puede contribuir al desarrollo de una muy buena aplicación de búsqueda de información en documentos y que, aunque ha sido probado únicamente como aplicación de escritorio, la fase de preprocesamiento nos permitiría extender dicha aplicación a otros ámbitos como la Web. Un escenario de prueba tan amplio como lo es la Web implica la realización de pruebas más densas ya que los grupos de datos disponibles en Internet tienen intersecciones más amplias, cambios constantes e incluso pertenencias indefinidas a algún grupo. El desarrollo de este tipo de pruebas queda propuesto como trabajo futuro.

Entre las recomendaciones que se dan en este trabajo de tesis está el realizar pruebas para utilizar

el algoritmo de cálculo local como un desambiguador de palabras y agregar adjetivos a los sustantivos para realizar consultas mucho más especificas sin perder la idea general de lo que se busca.

5.5 Trabajo futuro

En esta tesis se han empleado diversas técnicas de minería de datos para el desarrollo del modelo propuesto. Si bien los resultados obtenidos han sido satisfactorios, han quedado abiertas algunas líneas de investigación:

- Utilizar el método propuesto para otros problemas de minería de datos, como la desambiguación de palabras y el clustering de texto.
- La paralelización de algunos módulos del método y una propuesta para la construcción en paralelo del grafo.
- Procesamiento de documentos en otros idiomas, utilizando las diferentes bases de datos de WordNet.
- Probar la efectividad del método con documentos de Web y en formatos semánticos como OWL y RDF.
- Refinamiento de la prototipo de prueba, el cual podría utilizarse tanto como aplicación de escritorio y Web.
- Uso de adjetivos para formar relaciones del tipo adjetivo-sustantivo que permitan al usuario asignar una culidad a la informacion que está buscando.

A

Diagrama de clases de software de indexación y búsqueda

La Figura A.1 muestra el diagrama de clases para el proceso de extracción de texto.

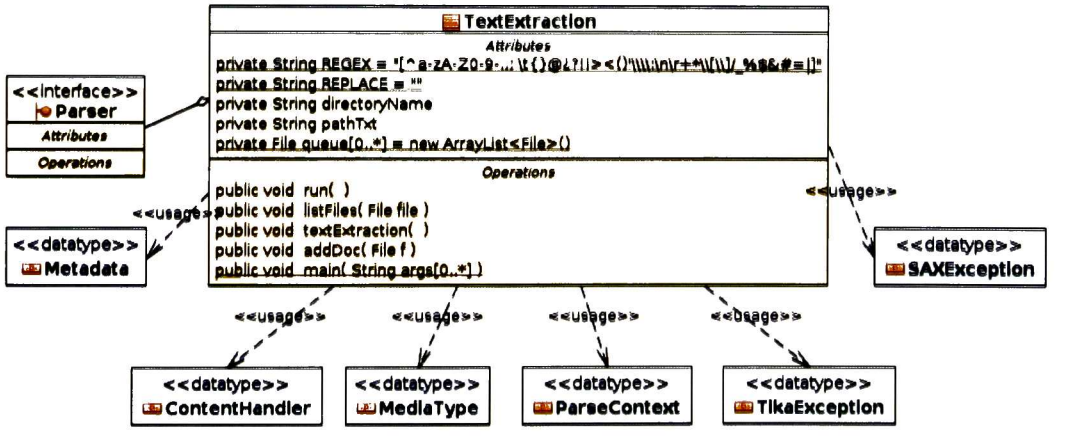


Figura A.1: Diagrama de clases para proceso de Extracción de Texto.

La Figura A.2 muestra el diagrama de clases para el Parser.

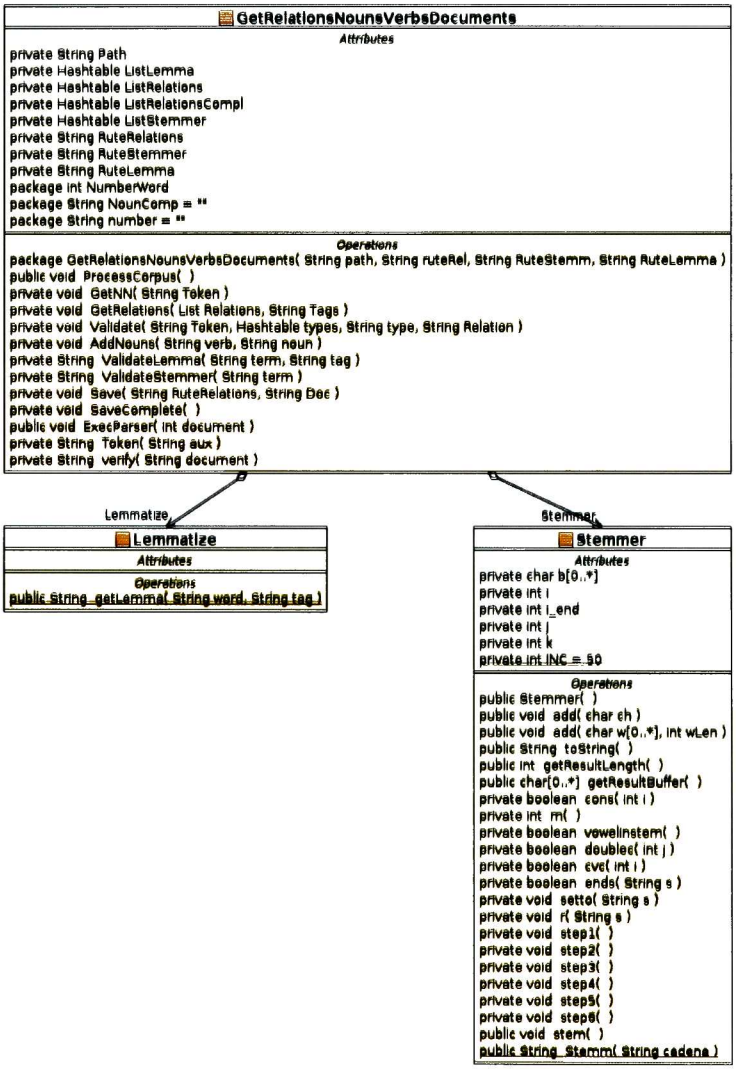


Figura A.2: Diagrama de clases para el Parser.

La Figura A.3 muestra el diagrama de clases para el cálculo de *tf-idf*.

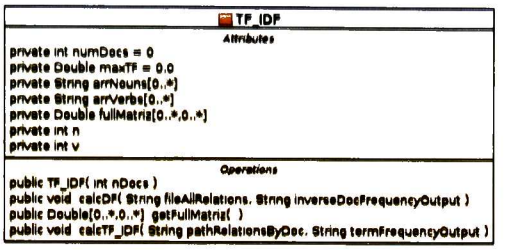


Figura A.3: Diagrama de clases para el cálculo de *tf-idf*.

La Figura A.4 muestra el diagrama de clases para JGibsLDA.

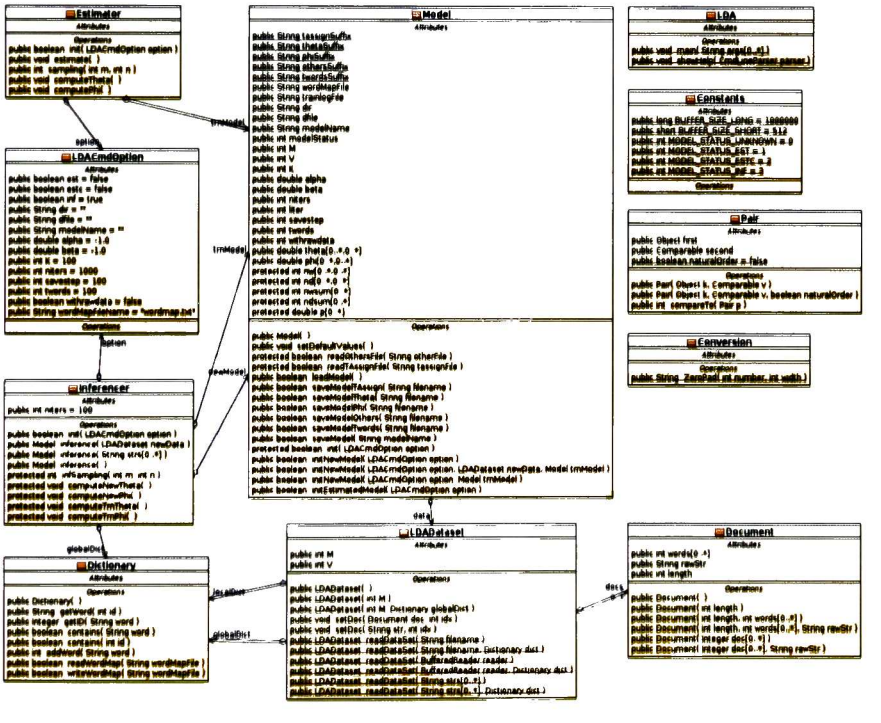


Figura A.4: Diagrama de clases para JGibsLDA.

La Figura A.5 muestra el diagrama de clases para la construcción y búsqueda del grafo.

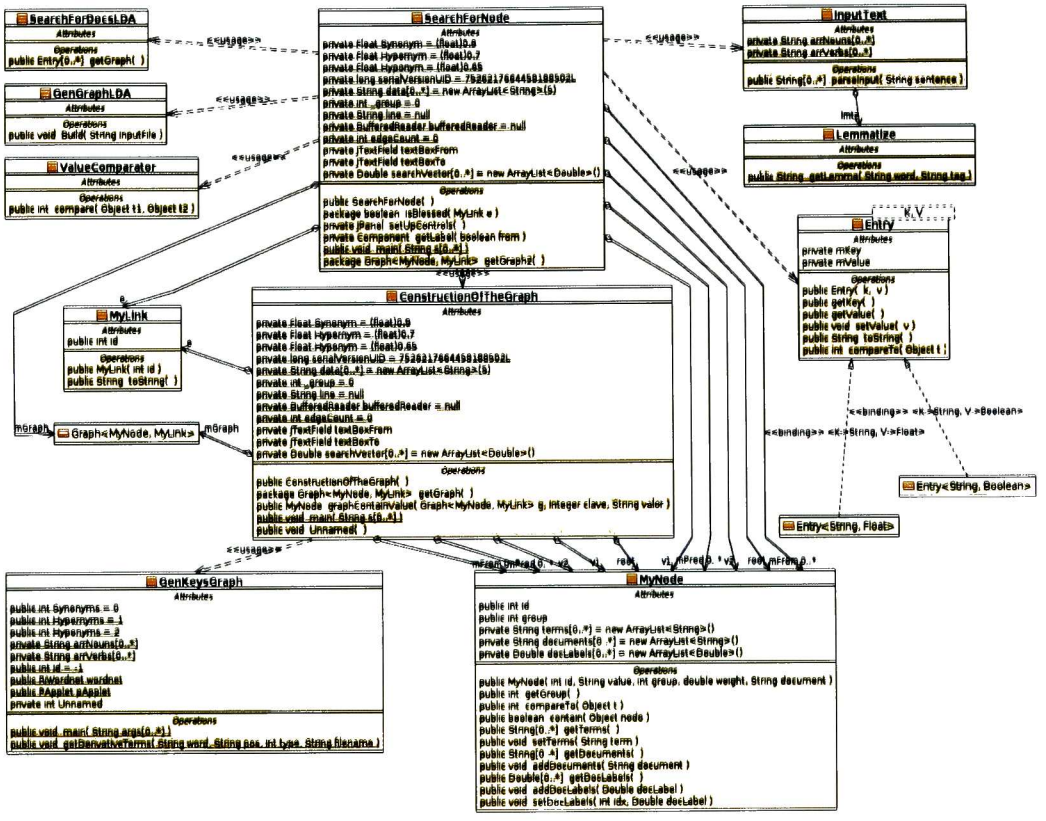


Figura A.5: Diagrama de clases para la construcción y búsqueda del grafo.

Bibliografía

[1] S. Al-Saffar and G. Heileman. Experimental bounds on the usefulness of personalized and topic-sensitive pagerank. In *Web Intelligence, IEEE/WIC/ACM International Conference on*, pages 671 –675, nov. 2007.

[2] Loulwah AlSumait and Carlotta Domeniconi. *Text Clustering with Local Semantic Kernels*. Springer London, 2008. 10.1007/978-1-84800-046-9_5.

[3] R. Amarasiri, D. Alahakoon, K. Smith, and M. Premaratne. Hdgsomr: a high dimensional growing self-organizing map using randomness for efficient web and text mining. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 215–221, sept. 2005.

[4] P.J. Antony, S.P. Mohan, and K.P. Soman. Svm based part of speech tagger for malayalam. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*, pages 339–341, march 2010.

[5] Antonio Badia. Personal information management for intelligence tasks. In Hsinchun Chen and Christopher Yang, editors, *Intelligence and Security Informatics*, volume 135 of *Studies in Computational Intelligence*, pages 215–226. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-69209-6_12.

[6] Ricardo Baeza-Yates. Challenges in the interaction of information retrieval and natural language processing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2945 of *Lecture Notes in Computer Science*, pages 445–456. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-24630-5_55.

[7] Richard Beckwith, George A. Miller, and Randee Tengi. Design and implementation of the wordnet lexical database and searching software. 1993.

[8] Ofer Bergman, Ruth Beyth-Marom, Rafi Nachmias, Noa Gradovitch, and Steve Whittaker. Improved search engines and navigation preference in personal information management. *ACM Trans. Inf. Syst.*, 26:20:1–20:24, October 2008.

[9] R. Besancon, M. Rajman, and J.-C. Chappelier. Textual similarities based on a distributional approach. In *Database and Expert Systems Applications, 1999. Proceedings. Tenth International Workshop on*, pages 180 –184, 1999.

[10] Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. Hybrid search: effectively combining keywords and semantic searches. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, ESWC'08, pages 554–568, Berlin, Heidelberg, 2008. Springer-Verlag.

[11] Yevgeniy Bodyanskiy, Yevgen Gorshkov, Vitaliy Kolodyazhniy, and Andreas Stephan. Combined learning algorithm for a self-organizing map with fuzzy inference. In Bernd Reusch, editor, *Computational Intelligence, Theory and Applications*, volume 33 of *Advances in Soft Computing*, pages 641–650. Springer Berlin / Heidelberg, 2005. 10.1007/3-540-31182-3_59.

[12] Tristan Blanc Brude and Dominique L. Scapin. What do people recall about their documents?: implications for desktop search tools. In *Proceedings of the 12th international conference on Intelligent user interfaces*, IUI '07, pages 102–111, New York, NY, USA, 2007. ACM.

[13] P Carr and Dilip B Madan. Option valuation using the fast fourier transform. *Journal of Computational Finance*, 2(4):61–73, 1999.

[14] S. Chawla and P. Bedi. Query expansion using information scent. In *Information Technology, 2008. ITSim 2008. International Symposium on*, volume 3, pages 1 –8, aug. 2008.

[15] Chun-Ling Chen, Frank Tseng, and Tyne Liang. An integration of fuzzy association rules and wordnet for document clustering. 5476:147–159, 2009. 10.1007/978-3-642-01307-2_16.

[16] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Int. Res.*, 24:305–339, August 2005.

[17] Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[18] Matteo Dimai and Nicola Torelli. Clustering textual data by latent dirichlet allocation: Applications and extensions to hierarchical data. In Francesco Palumbo, Carlo Natale Lauro, and Michael J. Greenacre, editors, *Data Analysis and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 241–248. Springer Berlin Heidelberg, 2010. 10.1007/978-3-642-03739-9_28.

[19] Xian Ding and Xinke Li. An ontology-based semantic expansion search model using semantic condition transform. In *Knowledge Acquisition and Modeling, 2009. KAM '09. Second International Symposium on*, volume 1, pages 359 –362, 30 2009-dec. 1 2009.

[20] Son Doan, Quang-Thuy Ha, and Susumu Horiguchi. A general fuzzy-based framework for text representation and its application to text categorization. volume 4223, 2006.

[21] Hans Fischer and Hans Fischer. Conclusion: The central limit theorem as a link between classical and modern probability theory. In J. Z. Buchwald, J. Lützen, and Gerald J. Toomer, editors, *A History of the Central Limit Theorem*, Sources and Studies in the History of Mathematics and Physical Sciences, pages 353–362. Springer New York, 2011. 10.1007/978-0-387-87857-7_8.

[22] Yan Fu, Dongqing Yang, Shiwei Tang, Tengjiao Wang, and Aiqiang Gao. *A New Text Clustering Method Using Hidden Markov Model*, volume 4592 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-73351-5_7.

[23] M. Golemati, A. Katifori, E.G. Giannopoulou, I. Daradimos, and C. Vassilakis. Evaluating the significance of the windows explorer visualization in personal information management browsing tasks. In *Information Visualization, 2007. IV '07. 11th International Conference*, pages 93–100, july 2007.

[24] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics*. Springer, Dordrecht, 2. edition, 2004.

[25] Zellig Harris. *Mathematical Structures of Language*. John Wiley and Son, New York, 1968.

[26] T.H. Haveliwala. Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):784 – 796, july-aug. 2003.

[27] Gregor Heinrich. Parameter estimation for text analysis. *Bernoulli*, pages 1–31, 2005.

[28] Sarah Henderson. Personal digital document management. In Masood Masoodian, Steve Jones, and Bill Rogers, editors, *Computer Human Interaction*, volume 3101 of *Lecture Notes in Computer Science*, pages 651–655. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-27795-8_72.

[29] William R. Hersh. *Information retrieval: a health and biomedical perspective*. Springer, third edition edition, 2009.

[30] H. Hu, X.Y. Du, D.Y. Liu, and J.H. Ouyang. *Ontology Learning Using WordNet Lexicon*. Springer Netherlands, 2006. 10.1007/978-1-4020-3953-9_36.

[31] Zhang Ji-Lin, Ren Yong-jian, Zhang Wei, Xu Xiang-Hua, Wan Jian, and Weng Yu. Webs ranking model based on pagerank algorithm. In *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pages 4811 –4814, dec. 2010.

[32] Liping Jing, Michael Ng, and Joshua Huang. Knowledge-based vector space model for text clustering. *Knowledge and Information Systems*, 25:35–55, 2010. 10.1007/s10115-009-0256-5.

[33] A. Katifori, E. Torou, C. Halatsis, G. Lepouras, and C. Vassilakis. A comparative study of four ontology visualization techniques in protege: Experiment setup and preliminary results. In *Information Visualization, 2006. IV 2006. Tenth International Conference on*, pages 417–423, july 2006.

[34] Ishwinder Kaur and Anthony J. Hornof. A comparison of lsa, wordnet and pmi-ir for predicting user click behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 51–60, New York, NY, USA, 2005. ACM.

[35] T. Kohonen, J. Hynninen, J. Kangas, and J. Lanksonen. Som-pak: the self-organizing map program package. Technical report, 1995.

[36] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982. 10.1007/BF00337288.

[37] Gerald Kowalski and Mark Maybury. *Introduction to Information Retrieval Systems*, volume 8 of *The Information Retrieval*. Springer Netherlands, 2002. 10.1007/0-306-47031-4_1.

[38] Man Lan, Chew Lim Tan, Jian Su, and Hwee Boon Low. Text representations for text categorization: A case study in biomedical domain. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 2557–2562, aug. 2007.

[39] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.

[40] Jianqiang Li, Yu Zhao, and Bo Liu. Fully automatic text categorization by exploiting wordnet. In *Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology*, AIRS '09, pages 1–12, Berlin, Heidelberg, 2009. Springer-Verlag.

[41] Ying Liu, Peter Scheuermann, Xingsen Li, and Xingquan Zhu. Using wordnet to disambiguate word senses for text classification. In *Proceedings of the 7th international conference on Computational Science, Part III: ICCS 2007*, ICCS '07, pages 781–789, Berlin, Heidelberg, 2007. Springer-Verlag.

[42] P. Malo, P. Siitari, O. Ahlgren, J. Wallenius, and P. Korhonen. Semantic content filtering with wikipedia and ontologies. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 518–526, dec. 2010.

[43] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[44] Thomas P. Minka. Estimating a Dirichlet distribution. 2003.

[45] Isidra Ocampo-Guzmán. Enfoque basado en datos para el aprendizaje de ontologías. Master's thesis, Cinvestav - Tamaulipas, Cd. Victoria, Tamaulipas, México., Enero 2010.

[46] Chris D. Paice. Stemming. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 2790–2793. Springer US, 2009.

[47] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using pagerank to characterize web structure. In Oscar Ibarra and Louxin Zhang, editors, *Computing and Combinatorics*, volume 2387 of *Lecture Notes in Computer Science*, pages 1–4. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-45655-4_36.

[48] Patrick Pantel and Dekang Lin. Efficiently clustering documents with committees. In Mitsuru Ishizuka and Abdul Sattar, editors, *PRICAI 2002: Trends in Artificial Intelligence*, volume 2417 of *Lecture Notes in Computer Science*, pages 527–535. Springer Berlin / Heidelberg, 2002.

[49] J.M. Park, J.H. Nam, Q.P. Hu, and H.W. Suh. Product ontology construction from engineering documents. In *Smart Manufacturing Application, 2008. ICSMA 2008. International Conference on*, pages 305–310, april 2008.

[50] Christos Pateritsas and Andreas Stafylopatis. A nearest features classifier using a self-organizing map for memory base evaluation. In Stefanos Kollias, Andreas Stafylopatis, Wlodzislaw Duch, and Erkki Oja, editors, *Artificial Neural Networks – ICANN 2006*, volume 4132 of *Lecture Notes in Computer Science*, pages 391–400. Springer Berlin / Heidelberg, 2006.

[51] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 91–100, New York, NY, USA, 2008. ACM.

[52] Luca Pretto. A theoretical analysis of google's pagerank. In Alberto Laender and Arlindo Oliveira, editors, *String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 125–136. Springer Berlin / Heidelberg, 2002. 10.1007/3-540-45735-6_13.

[53] Pamela Ravasio, Sissel Guttormsen Schär, and Helmut Krueger. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM Trans. Comput.-Hum. Interact.*, 11:156–180, June 2004.

[54] Santosh Kumar Ray, Shailendra Singh, and Bhagwati P. Joshi. Exploring multiple ontologies and wordnet framework to expand query for question answering system. In Uma Shanker Tiwary, Tanveer J. Siddiqui, M. Radhakrishna, and M. D. Tiwari, editors, *Proceedings of the First International Conference on Intelligent Human Computer Interaction, IHCI 2009, January 20-23, 2009, Organized by the Indian Institute of Information Technology, Allahabad, India*, pages 296–305. Springer India, 2009.

[55] Leo Sauermann and Dominik Heim. Evaluating long-term use of the gnowsis semantic desktop for pim. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 467–482, Berlin, Heidelberg, 2008. Springer-Verlag.

[56] H.L. Shashirekha and S. Murali. Ontology based structured representation for domain specific unstructured documents. In *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on*, volume 1, pages 50–54, dec. 2007.

[57] Shady Shehata. A wordnet-based semantic model for enhancing text clustering. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, ICDMW '09, pages 477–482, Washington, DC, USA, 2009. IEEE Computer Society.

[58] Bin Shi, Liying Fang, Jianzhuo Yan, Pu Wang, and Chen Dong. Classification of semantic documents based on wordnet. In *Proceedings of the 2009 International Conference on E-*

Learning, E-Business, Enterprise Information Systems, and E-Government, EEEE '09, pages 173–176, Washington, DC, USA, 2009. IEEE Computer Society.

[59] Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. Block-based similarity search on the web using manifold-ranking. In Karl Aberer, Zhiyong Peng, Elke Rundensteiner, Yanchun Zhang, and Xuhui Li, editors, *Web Information Systems – WISE 2006*, volume 4255 of *Lecture Notes in Computer Science*, pages 60–71. Springer Berlin / Heidelberg, 2006. 10.1007/11912873_9.

[60] Hongsheng Wang, Jiuying Qin, and Hong Shao. Expansion model of semantic query based on ontology. In *Web Mining and Web-based Application, 2009. WMWA '09. Second Pacific-Asia Conference on*, pages 86 –90, june 2009.

[61] James Z. Wang and William Taylor. Concept forest: A new ontology-assisted text document similarity measurement method. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, WI '07, pages 395–401, Washington, DC, USA, 2007. IEEE Computer Society.

[62] Jinghua Wang, Jianyi Liu, and Cong Wang. Keyword extraction based on pagerank. In Zhi-Hua Zhou, Hang Li, and Qiang Yang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 4426 of *Lecture Notes in Computer Science*, pages 857–864. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-71701-0_95.

[63] Yi Wang. Distributed Gibbs Sampling of Latent Dirichlet Allocation : The Gritty Details. Technical report, 2007.

[64] D. Widdows and T. Cohen. The semantic vectors package: New algorithms and public tools for distributional semantics. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 9 –15, sept. 2010.

[65] Dominic Widdows and Kathleen Ferraro. Semantic Vectors: a Scalable Open Source Package and Online Technology Management Application. In *Proceedings of the Sixth International*

Language Resources and Evaluation (LREC'08), Marrakech, Morocco, May 2008. European Language Resources Association (ELRA).

[66] Michael Wünstel, Daniel Polani, Thomas Uthmann, and Jürgen Perl. Behavior classification with self-organizing maps. *RoboCup 2000: Robot Soccer World Cup IV*, 2019/2001:108–118, 2001.

[67] Mingmin Xu, Liang He, and Xin Lin. A refined tf-idf algorithm based on channel distribution information for web news feature extraction. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 2, pages 15–19, march 2010.

[68] Hsin-Chang Yang and Chung-Hong Lee. Automatic category generation for text documents by self-organizing maps. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 3, pages 581–586, 2000.

[69] Yuangang Yao, Lanfen Lin, and Jinxiang Dong. Research on ontology-based multi-source engineering information retrieval in integrated environment of enterprise. In *Interoperability for Enterprise Software and Applications China, 2009. IESA '09. International Conference on*, pages 277–282, april 2009.

[70] Jun Ye. Multicriteria group decision-making method using vector similarity measures for trapezoidal intuitionistic fuzzy numbers. *Group Decision and Negotiation*, pages 1–12, 2010. 10.1007/s10726-010-9224-4.

[71] Hua-Jun Zeng, Xuan-Hui Wang, Zheng Chen, Hongjun Lu, and Wei-Ying Ma. Cbc: clustering based text classification requiring minimal labeled data. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 443–450, nov. 2003.

[72] Feng Zhonghui, Shen Junyi, and Bao Junpeng. An incremental algorithm of text clustering based on semantic sequences. *Wuhan University Journal of Natural Sciences*, 11:1340–1344, 2006. 10.1007/BF02829263.

[73] Guo-Song Zhou, Hong-Liang Li, Zheng-Ning Wang, and Guang-Hui Liu. An efficient intra prediction method for h.264 based on probability and statistics. In *Computational Problem-Solving (ICCP), 2010 International Conference on*, pages 413 –415, dec. 2010.

[74] Guobing Zou, Bofeng Zhang, Yanglan Gan, and Jianwen Zhang. An ontology-based methodology for semantic expansion search. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, volume 5, pages 453 –457, oct. 2008.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN

UNIDAD TAMAULIPAS

Cd. Victoria, Tamaulipas, a 14 de Diciembre de 2011

Los abajo firmantes, integrantes del jurado para el examen de grado que sustentará la C. ERIKA VELAZQUEZ GARCIA, declaramos que hemos revisado la tesis titulada:

"Organización Semántica de Documentos Mediante Grafos"

Y consideramos que cumple con los requisitos para obtener el grado de Maestro en Ciencias en Computación.

Atentamente,

Dr. Iván López Arévalo

Dr. Víctor Jesús Sosa Sosa

Dr. Javier Rubio Loyola

Km. 5.5 Carretera a Soto la Marina. Ciudad Victoria, Tamaulipas. CP 87130
Parque Científico y Tecnológico TECNOTAM
Tels.: (834) 1070220 y 316 6600
E-mail: admin@tamps.cinvestav.mx