Instituto Politécnico Nacional

Centro de Investigación en Computación

Secretaría de Investigación y Posgrado

Construcción de un árbol de términos latentes y su uso en el cálculo de la semejanza de documentos

T E S I S

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

EL ING. JUAN MONTOYA PÉREZ

DIRECTORES DE TESIS:
Dr. GILBERTO MARTINEZ LUNA
Dr. ADOLFO GUZMÁN ARENAS

MÉXICO, D.F. **2012**

Los problemas en la recuperación de información, se centran en cómo se estructura, guarda y recupera información de forma automática. Este trabajo esta centrado, en extender uno de los modelos clásicos para la recuperación de información, el Modelo Vectorial. De forma general esta extensión se resume en: descubrir términos latentes, guardarlos en una estructura de datos complementaria al archivo inverso (árbol de términos latentes) y usar este árbol de términos latentes para calcular la semejanza entre documentos.

Dedico a ustedes que vendrán;
/** extensión y mejora de los que hoy, somos proyecto inconcluso y perfectible de ustedes que estuvieron.
Porque en ustedes esto vivo seguirá, esto que en quienes estuvieron y en quienes estamos vive, esto que nos hace ser. */

Agradezco a ustedes que estuvieron;
/** a través de quienes hoy somos y estamos, pues mediante nosotros son, y en ustedes que estarán seguiremos siendo. */

Con sentimiento de esperanza al dedicar y de orgullo al agradecer;
/** Porque estoy puedo escribir y porque soy puedo sentir. */

Reconocimiento especial al Consejo Nacional de Ciencia y Tecnología (Conacyt), por la beca otorgada para realizar este trabajo, a través del Programa Nacional de Becas para Estudios de Posgrado.

CONTENIDO

Índice de Tablas:

Índice de Figuras y Gráficas:

Índice de Ecuaciones:

Resumen

La Web se convierte en un repositorio universal, del conocimiento y cultura humana, que a permitido la publicación de ideas e información en una escala nunca antes vista. Este gran universo atrae la atención de millones de personas, a buscar información útil en la Web, aunque esta tarea frecuentemente es tediosa y difícil. Estas dificultades, han atraído el interés renovado en las técnicas de Recuperación de información (RI), como soluciones prometedoras.

Los problemas en la recuperación de información, se centran en cómo se estructura, guarda y recupera información de forma automática. Este trabajo esta centrado, en extender uno de los modelos clásicos para la recuperación de información, el Modelo Vectorial. De forma general esta extensión se resume en: descubrir términos latentes, guardarlos en una estructura de datos complementaria al archivo inverso (árbol de términos latentes) y usar este árbol de términos latentes para calcular la semejanza entre documentos.

El descubrimiento de términos latentes, se involucra con la forma de interpretar la estructura de la información, la construcción de un árbol de términos latentes, es una alternativa para guardar la información de documentos y la forma de navegar y usar el árbol de términos latentes, para calcular la semejanza de documentos; es una alternativa útil para recuperar información de un conjunto de documentos, el trabajo trata de una extensión del modelo clásico vectorial, para la recuperación de información.

Palabras Clave: Término Latente, Árbol de términos latentes, Modelo Vectorial, Recuperación de Información

Clasificación ACM:

H. Information Systems/H.3 Information Storage and Retrieval/ H.3.1 Dictionaries
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.2 File organization
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.3 Query formulation
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.3 Retrieval models

Abstract

The Web becomes a universal repository of knowledge and human culture that has allowed the publication of ideas and information on a scale never seen before. This great universe attracts the attention of millions of people find useful information on the Web, although this task is often tedious and difficult. These difficulties have attracted renewed interest in the techniques of information retrieval (IR) as promising solutions.

Problems in information retrieval are focus on how to structure, store and retrieve information automatically. This work is focused on extend one of the classic models for information retrieval, the vectorial model. Overview this area is summarized in discover "latent terms", save them into a data structure complementary to the posting-list (latent terms tree) and use this tree to calculate similarity between documents.

The discovery of latent terms involves with, how to interpret the information structure, building a tree of latent terms is an alternative to store information about documents, and how to navigate and use the latent terms tree to calculate the similarity of documents is a useful alternative to retrieve information from a set of documents, together is an extension of classical vectorial model for information retrieval.

Keywords: Latent Term, Latent Term Tree, Vectorial Model, Information Retrieval

ACM Clasification:

H. Information Systems/H.3 Information Storage and Retrieval/ H.3.1 Dictionaries
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.2 File organization
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.3 Query formulation
H. Information Systems/H.3 Information Storage and Retrieval/ H.3.3 Retrieval models

Capítulo I
Introducción

La diferencia entre construcción y creación es exactamente esto: una cosa construida puede ser querida sólo después de construida; pero una cosa creada es querida antes de que exista.

Gilbert Keith Chesternon
Prefacio a Dickens, Pickwick Papers

1.1 Antecedentes

La recuperación de información es un área de las ciencias de la computación en constante crecimiento, esto debido a que la actual cantidad de información al alcance de la gente, en comparación con la cantidad de información que una persona es capaz de asimilar, es mucho muy superior; una de las preocupaciones para quien busca información es no perder tiempo con documentos que no estén acorde con el tópico de su interés.

La finalidad de los modelos clásicos para la recuperación de información, como su nombre lo indica, es recuperar información útil para quien busca; uno de los problemas comúnmente presentes en la recuperación de información, se da al ordenar un conjunto de documentos poniendo en la cima al documento más relacionado con una frase, cadena de palabras o documento con el que se pretende describir el tópico de interés.

Para encontrar los documentos de interés para quien consulta, existen modelos clásicos para la recuperación de información, estos modelos tienen virtudes, ventajas y diferencias propias. Este trabajo se centra de manera específica en uno de los modelos clásicos para la recuperación de información, el modelo vectorial.

Específicamente este trabajo tiene como fin proponer una extensión al modelo vectorial, que le permita calcular la semejanza entre documentos, tomando en cuenta términos compuestos por más de una cadena de caracteres.

1.2 Planteamiento de problema

El presente trabajo de tesis se centra en uno de los modelos clásicos para la recuperación de información, el modelo vectorial, este modelo tiene diferencias con respecto a los otros modelos clásicos para la recuperación de información, que lo hacen diferente.

Las tres diferencias con respecto a los modelos booleano y probabilístico que en este trabajo se toman en cuenta son:

1) La equiparación parcial, esto es, la capacidad del modelo para ordenar los resultados de una búsqueda, basado en el grado de semejanza entre cada documento de la colección y la consulta, a diferencia del booleano que solo puede decir si un documento tiene o no a un término sin importar el número de ocurrencias o en el caso del probabilístico que necesita una hipótesis inicial, para dar resultados basado en la experiencia de usuarios anteriores.

2) La ponderación de los términos en los documentos, no limitándose a señalar la presencia o ausencia de los mismos, sino asignando a cada término en cada documento un número real que refleje su importancia en el documento.

3) Y la ponderación de los términos en la consulta, de manera que el usuario puede asignar pesos a los términos de la consulta, que reflejen la importancia de los mismos en relación a su necesidad informativa.

Así como el modelo vectorial tiene diferencias útiles, presenta también limitaciones. De forma específica este trabajo tiene como finalidad, solventar la limitación del modelo vectorial para soportar el uso de términos compuestos por más de una palabra.

El problema principal antes referido, es la incapacidad para tomar en cuenta el orden en que aparecen los términos en los documentos. El modelo vectorial, es un modelo de bolsa de palabras, es decir, dos documentos que al compararse resulten semejantes si los términos que se usaron para compararlos están diciendo algo diferente.

Por ejemplo el documento siguiente: "José es más rápido que Juan" y otro documento: "Juan es más rápido que José" comparados con un tercer documento: "más que rápido es Juan José" resultaran idénticos. Es decir, debido al concepto de término que para el modelo vectorial corresponde a grandes rasgos con un conjunto de caracteres aislados del resto, por un espacio en blanco, este modelo no discrimina entre frases que signifiquen cosas diferentes, pues es un modelo de bolsa de palabras.

Cabe hacer mención que este trabajo esta centrado en lenguajes, como el español y el inglés, en los que no se da la fusión de palabras como es el caso del alemán. La incapacidad para tomar en cuenta el orden en que aparecen los términos, se vería solventada con la expansión de una estructura de datos comúnmente usada en este modelo, trayendo como consecuencia el consumo excesivo de recursos computacionales.

La estructura antes referida, es el archivo inverso en el que se guarda un diccionario de términos, donde cada término es una referencia, que lista los documentos en los que tal término aparece y cada una de estas referencias a documentos, a su vez referencia una lista que indica las posiciones en que el término aparece en ese documento. Tomando en cuenta que cada término tiene longitud uno (está compuesto por una sola palabra), la forma de solventar esta incapacidad, sería hacer una familia de archivos inversos, donde la longitud de cada término fuera dese uno hasta la longitud total del documento más largo, permitiendo así, identificar cualquier término que apareciera en cualquier documento contenido en la colección o algún documento nuevo.

Construir una familia de archivos inversos con estas características, resultaría en un consumo excesivo de espacio en memoria y de tiempo para su generación (aunque resolvería el problema de términos con longitud superior a uno).Este es un problema que tiene la opción de ser solucionado con una estructura de datos más eficiente y compacta, que da nombre a este trabajo (Árbol de términos latentes).

Este trabajo propone, el uso de todos los términos que ocurran en la colección de documentos, con la justificación de no perder características sintácticas y semánticas de los documentos participantes, dando como resultado un incremento en la demanda de espacio invertido, para guardar las direcciones en que todos los términos ocurren, sin importar el número de apariciones de estos. El presente trabajo propone, no eliminar las denominadas palabras vacías del diccionario, para poder generar los términos con longitud superior a uno, sin perder el significado original.

Hasta ahora hay dos partidas porque ocuparse, primero el consumo de memoria, además por el tiempo extra de buscar y operar en ella que esto implica. Con la creación de un árbol de términos latentes, se propone disminuir el excesivo consumo de memoria, así de forma implícita, el tiempo dedicado se verá reducido también.

Entonces el problema se ve resumido a modificar el modelo vectorial, para permitirle el uso de términos compuestos por más de una palabra, donde sea tomado en cuenta el orden en que las palabras aparecen.

1.3 Objetivo General

Crear una extensión del modelo vectorial que tome en cuenta los términos compuestos por una o más palabras.

1.4 Objetivos Particulares

Incluir en el modelo vectorial el concepto "término latente", basándose en el concepto de signo lingüístico, para darle significado.

Descubrir términos latentes en una consulta deseada y que son usados para comparar contra la colección de documentos donde recuperar los deseados, con la cual construir una estructura de datos de la que se pueda seleccionar un subconjunto de los términos latentes si se desea.

Permitir la equiparación parcial, esto es, la capacidad del modelo para ordenar los resultados de una búsqueda, basado en el grado de semejanza entre cada documento de la colección y la consulta, pero a diferencia del modelo clásico, en esta extensión se podrá usar términos latentes en lugar de términos, dando como resultado un modelo extendido en el que se puede tomar en cuenta la morfología sintagmática de los términos usados en la comparación

1.5 Justificación

El modelo vectorial no es capaz de soportar combinaciones de términos y tomarlas como diferentes.

El presente trabajo de tesis tiene como propósito resolver esta deficiencia. En este camino por resolver el problema del modelo bolsa de palabras, es encontrando un problema extra: de consumo de recursos computacionales. A este consumo excesivo de recursos, se da respuesta con un manejo más económico, construyendo un árbol de términos latentes en lugar de una familia de archivos inversos. Los beneficios por esto, hacen del modelo vectorial y cualquier modelo basado en el modelo vectorial un modelo más poderoso, dando justificación al trabajo realizado.

1.6 Beneficios esperados

Lo que se pretende en este trabajo es extender al modelo vectorial, dando como resultado un modelo que sea sensible de forma implícita a la morfología sintagmática, descrita mediante el documento usado para comparar contra los documentos pertenecientes a la colección. Esto sin la necesidad de hacer un análisis concreto sobre la morfología sintagmática de los documentos usados en el ejercicio de comparación.

1.7 Límites

La principal limitación de esta propuesta de extensión al modelo vectorial, esta dada por la incapacidad de la misma para reconocer la temática a la que pertenece el grupo de documentos a tratar o un subconjunto de estos, es decir, esta extensión se ve limitada a tratar con colecciones de documentos con una temática común.

Una característica de la lengua, es su constante evolución, en esta extensión no hace falta una referencia fija para preguntar si un término tiene o no significado (por ejemplo un diccionario). Referencia que puede estar no actualizada, impidiendo que se utilice el término nuevo en el cálculo de la semejanza entre documentos.

Capítulo II
Marco teórico

La extracción de información es la tarea que se encarga de identificar descripciones de eventos, en textos en lenguaje natural y por consiguiente, extraer la información relacionada a dichos eventos.
Patwardham S. & Riloff E. "Learning Domain- Specific Information Extraction Patterns from the Web", Proceedings of the ACL 2006 Workshop on Information Extraction Beyond the Document, pp. 66-73

Resumen

En este capítulo se inicia con la h uperación de información, para después dar un paseo por las dificultades encontradas para llegar a un concepto concreto de

recuperación de información y se describe como se llega a un concepto dependiente del conocido concepto de recuperación de datos.

La recuperación de información es una tarea que cuenta con modelos clásicos para llevarse a cabo; para respaldar la decisión de extender al modelo clásico vectorial y no a otro, se hace una descripción general de los tres modelos clásicos para la recuperación de información. La finalidad de esta descripción general es dar al lector un panorama más concreto sobre el problema abordado, esta descripción general también es útil como escalón previo, que permita de manera más clara, describir la sutil pero útil diferencia entre el modelo clásico vectorial y la extensión de este, motivo del presente trabajo.

Este trabajo se centra en la comparación entre un documento y una colección de documentos, para obtener como resultado un subconjunto de la colección, donde el orden en que se presentan, es el resultado de ordenar al más semejante al principio y al menos semejante al final. Esta extensión, donde no se toma en cuenta la morfología sintagmática (al menos no de forma explícita) de los documentos pertenecientes a la colección, no es el primero usado para calcular la semejanza entre documentos. En este capitulo también se aborda la descripción del concepto frases maximales frecuentes, que tienen su propia manera de ponderar el peso de cada frase maximal, en el cálculo de la semejanza entre documentos, principal diferencia con respecto a el uso de términos latentes, cuando de calcular la semejanza entre documentos se trata.

2.1 Recuperación de información

Desde mediados del siglo XX se ha trabajado en el área de la Recuperación de información, teniendo como constante su relevancia continua y en notable aumento. Entre otros posibles factores desencadenantes de este efecto, destacan dos: en primer lugar, el crecimiento espectacular y constante de la web, con el aumento en el número de documentos digitales a disposición de los usuarios de la red. En segundo lugar, el cambio producido por internet, en los hábitos de los usuarios en las diversas modalidades de acceso a la información, lo que ha traído consigo un crecimiento en la demanda de los servicios que internet proporciona.

Desde que Berners-Lee inventó la World Wide Web en 1989 mientras trabajaba en el Centro Europeo para la Investigación Nuclear (CERN, actualmente Organisation Européenne pour la Recherche Nucléaire), el número de usuarios de la red ha sufrido una evolución imparable. Inicialmente fueron 50 personas las que en 1989 compartían páginas web, pero solo cinco años más tarde se estimaba en 16 millones el número de usuarios en todo el mundo. Al cabo de otros cinco años, en 2000, la cifra de usuarios asciende a 451 millones, y en marzo de 2011 se alcanzan dos mil setenta y dos millones de usuarios.

Este desarrollo ha provocado que el número de documentos disponible en la red haya sufrido una evolución semejante. Como es fácil de comprender con semejantes cifras, resulta imposible pensar siquiera, en catalogar o clasificar manualmente esta gigantesca cantidad de documentos.
En consecuencia, la Recuperación de información, entendida como el área de conocimiento a la que atañe la representación, el almacenamiento, el tratamiento y el acceso automatizados a los documentos o a sus sustitutos, surge como la única vía posible para tratar de controlar este volumen de información digital.

En cuanto al segundo factor enunciado al inicio, hemos visto actualmente miles de millones de usuarios empleando habitualmente la web, como un medio para acceder y consultar aquella información que precisan.

Ante semejante volumen de documentos, los buscadores se han convertido en una herramienta imprescindible, para discernir las pocas páginas que incluyen la información buscada, frente a los millones de páginas restantes que resultan irrelevantes.

Acostumbrados cada vez más al empleo de las nuevas tecnologías de la comunicación y a las técnicas de recuperación mediante la formulación de consultas (introduciendo habitualmente una o varias palabras del lenguaje natural). La Recuperación de información se ha ido convirtiendo en un campo de conocimiento, cada vez más necesario al que acudir en busca de soluciones automatizadas, no sólo cuando hablamos de búsqueda de información en internet, sino en el propio ámbito bibliotecario al facilitar la creación de productos y servicios acordes con las nuevas demandas de los usuarios.

Son muchos los enfoques que se han experimentado para abordar el objetivo esencial de la Recuperación de información (RI), esto es, la recuperación de todos los documentos relevantes y al mismo tiempo rechazar todos los documentos irrelevantes, ante la formulación de una consulta por parte del usuario: desde el modelo booleano (por tratarse de una de las vías más simples desde el punto de vista teórico) hasta la aplicación de técnicas de Inteligencia artificial, entre las que podemos citar las redes neuronales (RN), los algoritmos genéticos (AG) o el Procesamiento del lenguaje natural (PLN).

El significado de recuperación de información dentro de las ciencias de la computación es confuso, tal como Rijsbergen afirma “se trata de un término que suele ser definido en un sentido muy amplio”[RIJ,1999]. Esta ambigüedad da como resultado varios conjuntos con opiniones diferentes al momento de definir lo que representa el vocablo Information Retrieval.

Un conjunto de autores usa este término como sinónimo de recuperación de datos, basándose en la perspectiva dada por las bases de datos. Otro conjunto de autores expresan las diferencias que encuentran según su experiencia y conocimiento entre ambos conceptos, pero sin dar uno que no dependa del concepto de recuperación de datos. Un grupo más de autores, no toma mayor consideración al mencionar el concepto y pasa de largo para explicar, lo que él hace en este campo de las ciencias de la computación. Por último un cuarto grupo no se molesta en mencionar el concepto.

El primer grupo de definiciones están muy influenciadas por la tecnología computacional, cuya evolución ha inducido a considerar sinónimos ha ambos conceptos, llegando a olvidar que se puede recuperar información sin procedimientos computacionales (aunque no es lo más común hoy en día). Aún así, el frecuente y necesario empleo de una tecnología, no debe sustituir el adecuado uso de los conceptos terminológicos. Un claro ejemplo de este desacierto, es el Glosario de la Asociación de Bibliotecarios Americanos, que define el término “information retrieval” como recuperación de la información en primera acepción y como recuperación de datos en

una segunda acepción [ALA, 1983], considerando ambos términos sinónimos en Lengua Inglesa2. Igualmente, el Diccionario Mac Millán de Tecnología de la Información considera a la recuperación de información, como las "técnicas empleadas para almacenar y buscar grandes cantidades de datos y ponerlos a disposición de los usuarios" [LON, 1989].

Un segundo grupo de autores fijan diferencias. Meadow piensa que la recuperación de la información es "una disciplina que involucra la localización de una determinada información dentro de un almacén de información o base de datos" [MEA, 1992]. Este autor, implícitamente, establece que la recuperación de información se encuentra asociada con el concepto de selectividad, ya que la información específica ha de extraerse siguiendo algún tipo de criterio discriminatorio (selectivo por tanto). Pérez-Carballo y Strzalkowski redundan en esta tesis: "una típica tarea de la recuperación de información, es traer documentos relevantes desde una gran archivo en respuesta a una pregunta formulada y ordenarlos de acuerdo con su relevancia" [PER, 2000].

Del mismo modo, Grossman y Frieder indican que recuperar información es "encontrar documentos relevantes, no encontrar simples correspondencias a unos patrones de bits" [GRO, 1998]. Meadow considera que no es lo mismo la recuperación de información entendida como traducción del término inglés information recovery, que cuando se traduce el término information retrieval, ya que "en el primer caso no es necesario proceso de selección alguno" [MEA, 1992].
De similar opinión es Blair, quien dedica gran parte de la presentación de su libro 'Language and representation in information retrieval' a asentar las diferencias entre data retrieval e information retrieval, utilizando como criterios distintivos, entre otros [BLA, 1990]:

1. En recuperación de datos se emplean preguntas altamente formalizadas, cuya respuesta es directamente la información deseada. En cambio, en recuperación de información las preguntas resultan difíciles, de trasladar a un lenguaje normalizado (aunque existen lenguajes para la recuperación de información, son de naturaleza mucho menos formal que los empleados en los sistemas de bases de datos relacionales) y la respuesta será un conjunto de documentos que probablemente contendrá lo deseado, con un evidente factor de indeterminación.

2. Según la relación entre el requerimiento al sistema y la satisfacción de usuario, la recuperación de datos es determinista y la recuperación de información es posibilista, debido al nivel de incertidumbre presente en la respuesta.

3. Éxito de la búsqueda. En recuperación de datos el criterio a emplear es la exactitud de lo encontrado, mientras que en recuperación de información, el criterio de valor es el grado en el que la respuesta satisface las necesidades de información del usuario, es decir, su percepción personal de utilidad.

Tramullas Saz destaca un aspecto importante de las reflexiones de Blair, la importancia, en ocasiones ignorada, que tiene el factor de predicción por parte del usuario, ya que éste debe intuir, en numerosas ocasiones, los términos que han sido utilizados para representar el contenido de los documentos, independientemente de la presencia de mecanismos de control terminológico.

Este criterio, “es otro de los elementos que desempeñan un papel fundamental en el complejo proceso de la recuperación de información” [TRA, 1997] y que no se presenta en el campo de la recuperación de datos.

Baeza-Yates plantea las diferencias entre ambos tipos de recuperación, con argumentos quizá algo menos abstractos que los anteriormente empleados por otros autores, destacando que “los datos se pueden estructurar en tablas, árboles, etc. para recuperar exactamente lo que se quiere, el texto no posee una estructura clara y no resulta fácil crearla” [BAE, 1999].
Para este autor, el problema de la recuperación de información se define como, “dada una necesidad de información (consulta + perfil del usuario + ...) y un conjunto de documentos, ordenar los documentos de más a menos relevantes para esa necesidad y presentar un subconjunto de aquellos de mayor relevancia”. En la solución de este problema se identifican dos grandes etapas:

1. Elección de un modelo que permita calcular la relevancia de un documento frente a una consulta.

2. Diseño de algoritmos y estructuras de datos que implementen este modelo de forma eficiente.

Baeza-Yates se preocupa especialmente de las estructuras de datos y métodos de acceso a los mismos [BAE, 1992], [BAE, 1999], siendo este autor una verdadera referencia en esta materia. Curiosamente, a la hora de definir la recuperación de información, en lugar de proponer una definición propia, emplea la elaborada por Salton: “la recuperación de la información tiene que ver con la representación, almacenamiento, organización y acceso a los ítem de información” [SAL, 1983].

Salton indica que, en principio, no deben existir limitaciones a la naturaleza del objeto informativo y Baeza-Yates incorpora la reflexión siguiente: “la representación y organización debería proveer al usuario un fácil acceso a la información en la que se encuentre interesado. Desafortunadamente, la caracterización de la necesidad informativa de un usuario no es un problema sencillo de resolver” [BAE, 1999].

El tercer grupo de autores emplea la definición formulada por Salton (base de la mayoría de definiciones de a bibliografía especializada), añadiendo el rasgo diferenciador, en que estos autores no profundizan en escrutar las diferencias entre “recuperación de datos” y “recuperación de información”; bien por no ser objeto de sus trabajos o por considerarlas suficientemente establecidas en trabajos previos. Feather y Storges ven a la recuperación de información como “el conjunto de actividades necesarias para hacer disponible la información a una comunidad de usuarios” [IEI, 1997].

Croft estima que la recuperación de información es “el conjunto de tareas mediante las cuales el usuario localiza y accede a los recursos de información que son pertinentes, para la resolución del problema planteado. En estas tareas desempeñan un papel fundamental los lenguajes documentales, las técnicas de resumen, la descripción del objeto documental, etc.” [CRO, 1987]. Tramullas Saz impregna su definición del carácter selectivo, comentado anteriormente al afirmar que “el planteamiento de la

recuperación de información en su moderno concepto y discusión, hay que buscarlo en la realización de los tests de Cranfield y en la bibliografía generada desde ese momento y referida a los mecanismos más adecuados para extraer, de un conjunto de documentos, aquellos que fuesen pertinentes a una necesidad informativa dada" [TRA, 1997].

El cuarto y último grupo de autores se distinguen porque eluden definir la recuperación de la información. Su máximo exponente Chowdhury, quien simplemente dedica el primer párrafo de su libro 'Introduction to modern information retrieval' a señalar que "el término recuperación de la información fue acuñado en 1952 y fue ganando popularidad en la comunidad científica de 1961 en adelante", mostrando después los propósitos, funciones y componentes de los SRI [CHO, 1999]. Otro autor perteneciente a esta corriente es Korfhage, quien se centra en el almacenamiento y recuperación de la información, considerando a estos procesos como las dos caras de una moneda. Para este autor, "un usuario de un sistema de información lo utiliza de dos formas posibles: para almacenar información en anticipación de una futura necesidad, y para encontrar información en respuesta a una necesidad" [KOR, 1997].

En este capítulo, se abordan los principios teóricos de los denominados modelos clásicos de RI: el booleano, el probabilístico y el vectorial, comentando al tiempo su vigencia en los sistemas de recuperación de información (SRI) actuales.

2.2 Modelos clásicos de Recuperación de información

2.2.1 Modelo booleano

Constituye el primer modelo teórico, el más antiguo, empleado para establecer el subconjunto de documentos relevantes, en relación a una consulta específica, de entre todos los que configuran la colección (ya se trate del fondo de una biblioteca o de todas las páginas disponibles en la web). Al mismo tiempo es, sin duda, uno de los más sencillos tanto desde un punto de vista teórico como práctico, al basarse en la teoría de conjuntos y en el álgebra de Boole además de ser fácil de diseñar e implementar en la práctica, por otra parte.

El procesamiento automatizado de un documento textual, comienza con la extracción de los términos de indización, es decir, los términos que van a ser utilizados para describir el contenido del documento.

La posibilidad más simple, consiste en considerar todas las palabras aisladas que aparecen en el texto como los términos de indización. Habitualmente se eliminan algunas (las denominadas palabras vacías, entre las que suelen figurar los números, las preposiciones, conjunciones, verbo ser, haber y estar), aunque la consideración o no de estos procesos añadidos (como la inclusión de una lista de palabras vacías) no influyen en absoluto sobre los principios teóricos del modelo.

Una vez extraídas las palabras del texto, se ordenan por orden alfabético y se guardan en un denominado fichero inverso, junto con la referencia del documento de donde proceden (normalmente un número de documento asignado previamente por el sistema). Si se repite este proceso con todos los documentos de la colección, obtendremos

finalmente un fichero inverso que almacena los siguientes datos:

En primer lugar, los términos de indización (las palabras) que aparecen en toda la colección (ya sean los propios textos, los resúmenes de los textos del fondo y/o los títulos).En segundo lugar, cada uno de dichos términos (palabras) incorpora una lista con los números de los documentos en los que aparece.

Conviene destacar en este proceso que no se ha guardado noticia alguna sobre la frecuencia de aparición de cada término en cada documento. De ahí que el modelo booleano clásico sea denominado modelo binario, pues de la consulta del fichero inverso únicamente puedo saber si un determinado término de indización está presente (en cuyo caso se simbolizará por el número 1) o está ausente (en cuyo caso se simbolizará por el número 0) en cada uno de los documentos de la colección.

De manera que el archivo inverso (en concreto el fichero diccionario) puede representarse por una tabla cuyos datos básicos son los siguientes:

	D1	D2	…	Dn
T1	0	0	…	1
T2	1	0	…	1
…	…	…	…	…
Tn	1	0	…	0

Tabla 2.1 Archivo inverso

Donde T1, T2, ..., "Tn" son los términos de indización empleados en la colección; D1, D2, ..., Dn son los documentos que componen la colección; y donde el "1" significa que el término correspondiente aparece en ese documento concreto, mientras que el "0" significa que el término no aparece en dicho documento. Ello implica que no se tiene en cuenta la frecuencia de aparición de los términos en los documentos: tanto si aparece veinte veces como si aparece una sola vez, en todos los casos ese término en dicho documento se representará mediante un "1", y "0" cuando no aparezca. Se comprende entonces la denominación de modelo binario que recibe, pues únicamente se juega con dos posibilidades: la aparición y la no aparición de los descriptores en los documentos.
Si observamos la tabla, podemos deducir de ella las dos representaciones empleadas al manejar el modelo binario.

Por otra parte, cada término de indización se representa por la lista de documentos en los que aparece, lo que implica la observación de la tabla por filas:

T1 = {D1, ..., Dn}
T2 = {D2, ...}
.....
Tt = {D2, ..., Dn}

Figura 2.1 Términos y listas de documentos

Por otra parte, cada documento se representa por la lista de ceros y unos correspondientes a los términos de indización que contiene, lo que implica la observación de la tabla por columnas:

D1 = {1, 0, ... , 0}
D2 = {0, 1, ..., 1}
.....
Dn = {1, 0, ..., 1}

Figura 2.2 Documentos y listas de documentos

Se comprende bien ahora por qué se dice que en el modelo binario, todo documento se representa mediante una serie ordenada de ceros y unos, tantos como términos de descripción se empleen en la colección: el primer número corresponderá siempre a T1, el segundo dígito corresponderá a T2, y así sucesivamente hasta llegar a Tt, siendo t el número de descriptores distintos que representan el contenido de esa colección.

La misma tabla nos servirá para explicar el método empleado por los SRI basados en este modelo para contestar a las consultas formuladas por los usuarios. En primer lugar, el usuario debe introducir palabras, precisamente aquéllas que describan su necesidad informativa, o una fórmula que se ajuste a la sintaxis booleana. Habitualmente los usuarios empleamos pocas palabras, de manera que muchos SRI presentan un número máximo de palabras posibles en la consulta que ronda la decena (Google, por ejemplo). Si el usuario se limita a introducir dos palabras, por ejemplo, el sistema automáticamente convertirá dicha consulta a una fórmula booleana, introduciendo entre las dos palabras una conectiva por defecto, habitualmente AND.

Con un ejemplo se comprenderá fácilmente el procedimiento seguido por un SRI. Supongamos que deseamos localizar documentos sobre "publicaciones de programación en el CIC". Lo que solemos hacer es introducir en la ventana de un buscador en Internet esa frase tal cual. El SRI analiza la cadena de caracteres introducida en la ventana y la trata en principio, como si se tratase de un documento más. En consecuencia, eliminará las palabras vacías (imaginemos que "de", "en" y "el" lo son, lo que no resultaría extraño en español), resultando las palabras publicaciones, programación y CIC. A continuación, el SRI introduce entre ellas la conectiva por defecto, normalmente AND como ya dijimos. La fórmula en este caso resultaría:

Publicaciones AND programación AND CIC.

A continuación el sistema trata de localizar cada una de las palabras en su archivo inverso. Pueden suceder dos cosas: que figure o que no figure entre los términos de indización almacenados en el SRI en cada una de ellas:

Si figura, sustituye la palabra por el conjunto o lista de documentos de la colección donde aparece dicha palabra o término de indización.

Si no figura, lo que suele ser muy extraño en SRI cuyas colecciones abarcan millones de documentos de todo tipo, sustituye dicha palabra por el conjunto vacío (no aparece en ningún documento de la colección).

De ahí que cuando introducimos una sola palabra y ésta no aparece en el fichero inverso, y por lo tanto no se halla en ninguno de los documentos de la colección, un SRI basado en el modelo booleano muestre la siguiente respuesta: "No existe en la colección ningún documento que incluya dichas palabras".

Por último, con las listas que sustituyen a las palabras, el sistema efectúa las

operaciones de conjuntos correspondientes a las conectivas que figuren en la consulta, de la siguiente manera: El resultado de dos listas o conjuntos de documentos unidos por la conectiva AND, da como resultado el conjunto de los documentos en los que aparecen simultáneamente ambos términos.

El resultado de dos listas o conjuntos de documentos unidos por la conectiva OR, da como resultado, el conjunto de los documentos en que aparece el primer término (y no el segundo) o el segundo término (y no el primero) o ambos términos simultáneamente.
El resultado de una lista o conjunto de documentos, precedido por la conectiva NOT da como resultado el conjunto de los documentos de la colección en los que no aparece el término.

Por ejemplo, tras las operaciones de conjuntos correspondientes a la consulta: "publicaciones AND programación AND CIC", obtendríamos el conjunto de los documentos de la colección, en los que aparecen simultáneamente las palabras publicaciones, programación y CIC.

Una propiedad importante de los sistemas de recuperación, basados en el modelo booleano es que no pueden efectuar ningún proceso de ordenación, con los documentos resultantes de la búsqueda, pues todos ellos cumplen la fórmula en idénticas condiciones. Esta característica suele denominarse equiparación exacta, impidiendo que el sistema pueda situar en primer lugar aquel documento posiblemente más útil o relevante para el usuario y relegando a las últimas posiciones, a aquellos otros documentos con menos probabilidades de ser relevantes en relación a la consulta.

¿Cómo podría efectuar el sistema una ordenación con los documentos de la respuesta? Existen muchas posibilidades, pero una muy sencilla que permite comprender el proceso de clasificación, consistiría en ordenar los documentos por el número total de veces que aparece alguna de las palabras en ellos. Así, el primer documento podría contabilizar 45 apariciones (10 veces aparece "publicaciones", "programación" surge 30 veces y "CIC" aparece 5 veces, por ejemplo), el segundo documento contabilizaría 31 apariciones, el tercero incluiría 12 apariciones y así sucesivamente en orden decreciente.

Como podemos observar, el carácter binario (consideración exclusivamente de la presencia o ausencia de los términos en los documentos), es el principal responsable de la equiparación exacta, siendo considerado la principal desventaja del modelo. De hecho, la ponderación se ha demostrado muy útil para mejorar los resultados de la recuperación. Fácilmente podemos extender el modelo binario para permitir ponderación inexacta, usando por ejemplo la Distancia de Hamming, que se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida. Si dos palabras de código difieren en una distancia ***d***, se necesitan ***d*** errores para convertir una en la otra.

A pesar de esta desventaja, todavía hoy sigue constituyendo el modelo más fácil de utilizar por parte de un usuario medio (basta introducir palabras relativas a la necesidad informativa) y es bastante eficaz en los resultados obtenidos (en gran parte debido al volumen de documentación presente en la red, lo que provoca que la reducción de la respuesta a los documentos que satisfagan estrictamente las condiciones de la consulta por defecto, recordemos, la conectiva AND, aún genera subconjuntos muy abultados de documentos).

2.2.2 Modelo probabilístico

Introducido en la década de los setenta por Robertson y Sparck Jones [Sparck-R. & Robertson-S.], también es conocido como modelo de recuperación de independencia binaria (BIR). Este modelo se basa en las siguientes consideraciones:

Para caracterizar los documentos de la colección se han empleado ciertos términos de indización (palabras en principio).

Dada una necesidad informativa del usuario, existe un subconjunto de documentos de la colección, que contiene exclusivamente los documentos relevantes en relación a ella.
Si el usuario supiese los términos de indización que permiten caracterizar, tal subconjunto de documentos relevantes (porque aparecen en ellos y no aparecen en el resto de los documentos de la colección), tendríamos el problema resuelto. Como vemos, el modelo probabilístico parte exclusivamente de la presencia o ausencia de los términos en los documentos de la colección. Se trata, pues, también de un modelo binario, como el modelo booleano.

Lamentablemente, en un caso real el usuario no sabe cuáles son los términos de indización, que configurarían la consulta ideal. Tampoco sabe, de hecho, en qué medida los términos empleados en la consulta, permiten discernir los documentos relevantes y rechazar simultáneamente los documentos irrelevantes.

El modelo probabilístico actúa precisamente sobre los términos que configuran la consulta del usuario, ponderándolos; esto es, imponiéndoles un peso o número a cada uno de ellos, mayor cuanto mejor permita discernir los documentos relevantes de los irrelevantes, y menor en caso contrario. De esta manera, se persigue que el sistema efectúe la recuperación, incidiendo sobre todo en los mejores descriptores de entre los empleados por el usuario en la consulta, minimizando la importancia de aquellos otros términos que, aún figurando en la consulta, son malos descriptores del conjunto respuesta ideal.

Como tampoco se puede saber a priori, cuáles de entre los términos que configuran la consulta, son buenos descriptores y cuáles no lo son, a este modelo no le queda otro remedio que considerar, para cada uno de los términos empleados en la consulta, la "probabilidad de ser buen descriptor" (probabilidad de que el término empleado en la consulta, esté presente en un documento del conjunto de documentos relevantes en relación a la consulta) y simultáneamente, para ese mismo término, la "probabilidad de ser mal descriptor" (probabilidad de que ese mismo término, esté presente en un documento del conjunto de documentos irrelevantes en relación a la consulta).

Ahora bien, como estas probabilidades, para cada uno de los términos empleados en la consulta, son desconocidas en el momento de formalizar dicha consulta, este modelo se ve en la necesidad de efectuar una hipótesis inicial sobre sus valores. La obligatoriedad, de hacer una hipótesis inicial sobre las "probabilidades de ser buen y mal descriptor" para cada término de la consulta, se ha considerado el principal inconveniente de este modelo.

Basados en estos pesos iniciales, el modelo probabilístico es capaz de calcular el grado

de semejanza existente entre cada documento de la colección y la consulta ponderada, consiguiendo ordenar los documentos de la colección en orden descendente, de probabilidad de relevancia en relación a la consulta.

De esta manera el modelo probabilístico, supera el gran inconveniente puesto de manifiesto en el modelo booleano, la equiparación exacta. En efecto, el modelo probabilístico, aun siendo un modelo binario, efectúa equiparación parcial, lo que permite ordenar los documentos de la respuesta, conforme a su probabilidad de relevancia, ya que no puede ponderar los términos de la colección (es un modelo binario), la equiparación parcial es posible gracias a la ponderación de los términos empleados en la consulta.

Una de las grandes aportaciones del modelo probabilístico a la recuperación de información, consiste en el fenómeno denominado retroalimentación por relevancia. Aunque con carácter muy general consiste, en la utilización de información generada en procesos de recuperación anteriores, o durante el propio proceso de búsqueda, en el modelo probabilístico clásico consiste en mejorar los resultados de la recuperación, solicitando al usuario, tras la respuesta inicial del sistema, que analice los documentos recuperados (alrededor de la primera media docena) y juzgue cuáles son relevantes.

Con esta información se imponen nuevos valores a las "probabilidades de ser buen y mal descriptor", para cada término de la consulta, obteniéndose una nueva respuesta de documentos ordenados por su probabilidad de relevancia, aunque ahora mejorada, sin duda gracias a la información suministrada directamente por el usuario.

Actualmente son muchos los sistemas de recuperación de información que emplean alguna variante de la retroalimentación por relevancia, para mejorar y refinar los resultados de la búsqueda. Quizá la más conocida consista en la sugerencia al usuario de más resultados precedidos; es una manera de emplear la información precedente, en este caso, de procesos de recuperación anteriores.

Frente al modelo booleano, basado en teoría de conjuntos, y el modelo vectorial, de carácter algebraico, el modelo probabilístico formaliza el proceso de recuperación en términos de teoría de probabilidades. El objetivo perseguido en el modelo, es el de calcular la probabilidad de que un documento sea relevante para la consulta dado que dicho documento posee ciertas propiedades, (términos) que dicho documento contiene.

El rendimiento óptimo de un sistema se consigue cuando los documentos son ordenados de acuerdo a sus probabilidades de relevancia. En consecuencia, el sistema devolverá los documentos en orden decreciente de las probabilidades de relevancia estimadas mediante el modelo probabilístico.

El modelo parte de las siguientes suposiciones:

1. Todo documento es relevante o no relevante, para la consulta.
2. El hecho de juzgar un documento dado como relevante o no relevante, no aporta información alguna sobre la posible relevancia o no relevancia de otros documentos.

En base a ellas, y dada una consulta q, el modelo asigna a cada documento d_j, como medida de semejanza respecto a la consulta, el ratio $P(d_j \text{ relevante para } q)/P(d_j \text{ no relevante para } q)$, medida según la cual los documentos son devueltos, ordenadamente, al usuario.

Sea R, pues, el conjunto de documentos que sabemos (o hemos estimado) relevantes, y sea $\bar{R}$ su complementario (es decir, los no relevantes).
Sea $P(R| \rightarrow d_j)$ la probabilidad de que el documento d_j sea relevante para la consulta q, y $P(\bar{R}| \rightarrow d_j)$ la probabilidad de que el documento d_j sea no relevante para la consulta q.

La medida de semejanza $sem(d_j, q)$ del documento d_j respecto a la consulta q se define como el ratio:

$$sem(d, q) = \frac{P(R|d_j)}{P(\bar{R}|d_j)} \quad (2.1)$$

Que podemos descomponer en:

$$sem(d, q) = \frac{P(d_j|R) \times P(R)}{P(d_j|\bar{R}) \times P(\bar{R})} \quad (2.2)$$

donde $P(d_j|R)$ representa la probabilidad de seleccionar aleatoriamente el documento d_j de entre el conjunto de relevantes R y P(R), representa la probabilidad de que un documento de la colección sea relevante. Sus análogos y complementarios vienen dados por $P(d_j|\bar{R})$ y $P(\bar{R})$.
Puesto que $P(\bar{R})$ y $P(R)$ son constantes para todos los documentos de la colección, pueden ser simplificados obteniendo:

$$sem(d, q) \sim \frac{P(d_j|R)}{P(d_j|\bar{R})} \quad (2.3)$$

Asumiendo la independencia entre los términos índice se obtiene:

$$sem(d, q) \sim \frac{(\Pi_{g_i(d_j)=1} P(t_i|R) \times (\Pi_{g_i(d_j)=0} P(\overline{t_i}|R))}{(\Pi_{g_i(d_j)=1} P(t_i|\bar{R}) \times (\Pi_{g_i(d_j)=0} P(\overline{t_i}|\bar{R}))} \quad (2.4)$$

donde g_i es una función que devuelve el peso asociado al término k_i, dentro de un vector t-dimensional es decir, $g_i(d_j) = w_{i,j}$, con $w_{i,j} \in \{0,1\}$, $P(t_i|R)$, representa la probabilidad de que un documento seleccionado aleatoriamente de R contenga el término índice t_i, mientras que $P(\overline{t_i}|R)$ representa la probabilidad de que un documento seleccionado aleatoriamente de R, no contenga dicho término índice t_i. Las probabilidades asociadas con el conjunto R, tienen significados análogos. Tras aplicar logaritmos y eliminar algunos factores constantes dentro de una misma consulta, y sabiendo que $P(t_i|R) + P(\overline{t_i}|R) = 1$, obtenemos:

$$sem(d, q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times \left(\log \frac{P(t_i|R)}{1 - P(t_i|R)} + \log \frac{1 - P(t_i|\bar{R})}{P(t_i|\bar{R})} \right) \quad (2.5)$$

donde t es el número de términos que componen el vocabulario del sistema y los pesos de los términos son binarios, $w_{i,j} \in \{0,1\}$ y $w_{i,q} \in \{0,1\}$, indicando meramente la aparición o no del término, en el documento o consulta respectivamente.

Dado que inicialmente el conjunto R, no es conocido, se hace necesario estimar las probabilidades $P(t_i|R)$ y $P(t_i|\bar{R})$. De este modo, sea V un subconjunto de los documentos inicialmente devueltos y que es considerado relevante y sea V_i el subconjunto de V cuyos documentos, contienen el término t_i. Aproximando $P(t_i|R)$ mediante la distribución del término t_i en V:

$$P(t_i|R) = \frac{|V_i|}{|V|} \quad (2.6)$$

donde $|V|$ y $|V_i|$, representan el número de elementos en los conjuntos V y V_i, respectivamente. De forma similar y suponiendo que el resto de los documentos son no relevantes, aproximaremos $P(t_i|\bar{R})$ mediante:

$$P(t_i|\bar{R}) = \frac{n_i - |V_i|}{N - |V|} \quad (2.7)$$

Donde N es el tamaño de la colección de documentos y n_i, el número de documentos de la colección que contienen el término t_i. Para evitar problemas con valores pequeños de $|V|$ y $|V_i|$, comunes en la práctica, se introduce un factor de ajuste, obteniendo finalmente:

$$P(t_i|R) = \frac{|V_i| + 0{,}5}{|V| + 1} \quad (2.8)$$

$$P(t_i|\bar{R}) = \frac{n_i - |V_i| + 0{,}5}{N - |V| + 1} \quad (2.9)$$

Substituyendo dichas estimaciones en la expresión 2.8, obtenemos finalmente tras operar:

$$sem(d,q) \sim \sum_{i=1}^{t} w_{iq} \times w_{ij} \times w(1) \quad (2.10)$$

donde $w(1)$ es el denominado peso Robertson-Sparck Jones, de importancia clave en los esquemas de peso probabilísticos, que se define como:

$$w(1) = \log \frac{(|V_i| + 0{,}5)/(|V| - |V_i| + 0{,}5)}{(n_i - |V_i| + 0{,}5)/(N - n_i - |V| + |V_i| + 0{,}5)} \quad (2.11)$$

Múltiples medidas de semejanza basadas en esta expresión inicial, han sido empleadas en diversos sistemas, que junto al vectorial tf–idf, es punto de referencia para el desarrollo y evaluación de nuevos modelos y nuevos esquemas de pesos.

En un principio el modelo probabilístico supone 50% relevante y 50% no relevante a los descriptores nuevos en el sistema.

2.2.3 Modelo vectorial

Como hemos observado, se considera que el modelo probabilístico clásico, supera al modelo booleano clásico, en cuanto que el probabilístico efectúa equiparación parcial, mientras que el modelo booleano clásico efectúa equiparación exacta. Sin embargo, ambos siguen presentado una característica negativa: ni el modelo booleano ni el modelo probabilístico, tienen en cuenta la frecuencia con la que aparecen los términos de indización, dentro de los documentos. Esto es, ambos son modelos binarios de representación documental.

Parece lógico pensar que, si en un documento aparece el término "computadora" una vez, y en otro documento aparece ese mismo término veinte veces, consideremos que en el primer documento la importancia de "computadora" es menor que ese mismo término en el segundo documento. En consecuencia, surge un tercer modelo de recuperación, el modelo vectorial, basado en tres principios:

La equiparación parcial, esto es, la capacidad del sistema para ordenar los resultados de una búsqueda, basado en el grado de semejanza entre cada documento de la colección y la consulta.

La ponderación de los términos en los documentos, no limitándose a señalar la presencia o ausencia de los mismos, sino adscribiendo a cada término en cada documento un número real que refleje su importancia en el documento.

La ponderación de los términos en la consulta, de manera que el usuario puede asignar pesos a los términos de la consulta, que reflejen la importancia de los mismos en relación a su necesidad informativa.

Gracias a esta representación, tanto los documentos como las consultas pueden tratarse matemáticamente como vectores en un espacio ***t*** dimensional, de donde el modelo vectorial toma su nombre.

Para que podamos comprender las consecuencias de este hecho, consideraremos únicamente dos dimensiones (esto es, dos únicos términos). Sean por ejemplo, los documentos creados al apuntar las respuestas de María (**Figura 2.3**) que sólo sabe contestar con los monosílabos *si* y *no*.

D1={si si si si no}
D2={si no no no no}
C={no no no si si}

Figura 2.3 Documentos D1, D2 y *C*

Estos documentos se pueden representar en un gráfico como el de la **Figura 2.4** donde se representa a los documentos D1, D2 y C como vectores.

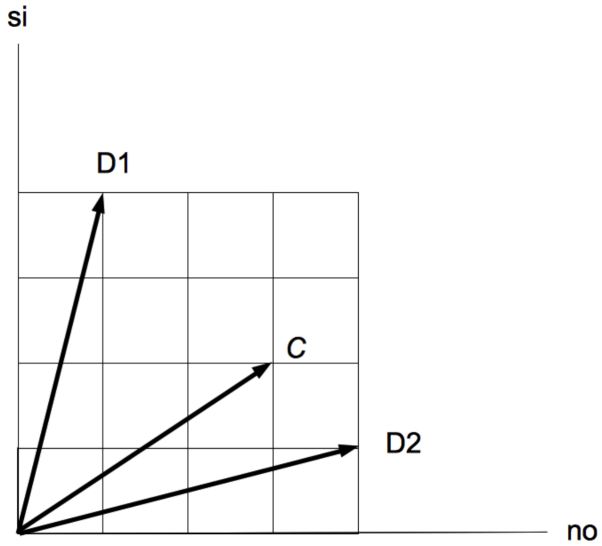


Figura 2.4 Representación vectorial de los documentos D1, D2 y *C*

D1=(4, 1) y D2=(1,4) y la consulta *C*=(2,3). Tratándose de un espacio bidimensional, podemos dibujar tanto los documentos como la consulta en el plano de esta hoja, como "flechas" o vectores que parten del origen de coordenadas, cuyo primer número corresponde al número de veces que se repite el término *si*, representado en el eje de abscisas y cuyo segundo número corresponde al número de veces que se repite, el término *no* representado en el eje de ordenadas.

Como podemos observar en el gráfico, puede resultar relativamente fácil juzgar cuál de los dos documentos se asemeja más a la consulta. Considerando que el vector de la consulta *C* está más próximo a D2, podemos deducir gráficamente que el orden de relevancia de los documentos D1 y D2 en relación a la consulta *C*, sería en el ejemplo: D2 y posteriormente D1. En resumen, en el modelo vectorial basta fijar un criterio de semejanza, para poder ordenar por relevancia, los documentos de una colección en relación a una consulta.

En cuanto a la manera de ponderar los términos en los documentos de la colección, una de las más utilizadas y de eficacia probada, es la denominada ponderación tf.idf. Consiste en multiplicar dos factores que reflejan la importancia de los términos, es decir, el número de ocurrencias de un término en un documento:

$$tf_{t,d} = número\ de\ ocurrencias\ del\ término\ t\ en\ el\ documento\ d \qquad (2.12)$$

El primer factor, $tf_{t,d}$ (abreviatura de Term Frequency), pretende reflejar la importancia de los términos en los documentos, concediendo mayor importancia a los términos cuantas más veces aparezcan en los documentos. La versión más sencilla de este factor, lo representa numéricamente mediante la frecuencia de aparición de cada término en cada documento de la colección.

$$N = número\ de\ documentos\ en\ la\ colección$$
$$n_t = número\ de\ documentos\ en\ la\ colección\ con\ el\ término\ t$$
$$idf_t = log\frac{N}{n_t} \qquad (2.13)$$

El segundo factor, idf_t (abreviatura de Inverse Document Frequency) o inverso de la frecuencia de documentos, pretende reflejar la importancia de los términos en la colección.

Así, se dará mayor importancia a un término cuanto menor sea el número de documentos de la colección, en los que aparezca dicho término. Por el contrario, si un término aparece en todos los documentos de la colección, su precisión y poder discriminatorio (capacidad para discernir los documentos relevantes de los irrelevantes ante una consulta) es nulo (tal término aparecerá necesariamente tanto en todos los documentos relevantes como en todos los documentos irrelevantes), de manera que se le otorgará una importancia mínima en esa colección en concreto (puede que en otra colección ese mismo término posea una gran importancia, porque aparece en muy pocos documentos).

Suele representarse numéricamente de manera proporcional, al logaritmo natural del inverso, del número de documentos de la colección en los que aparece dicho término.
Como se expuso anteriormente, el modelo vectorial, propone evaluar el grado de semejanza entre los documentos de una colección y las consultas, mediante algún criterio que muestre la mayor o menor cercanía entre los vectores correspondientes a los documentos y el vector correspondiente a la consulta.

$$\overline{A} \cdot \overline{B} = \left|\overline{A}\right|\left|\overline{B}\right| \cos(\theta) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (2.14)$$

Una de las maneras más habituales de cuantificar el nivel de cercanía entre vectores, es mediante el coseno del ángulo que forman, pues presenta la propiedad de ser un número mayor cuanto más cercanos estén entre sí ambos vectores (en el límite, el coseno de 0 vale la unidad), mientras que es un número menor cuanto más alejados estén entre sí (en el límite, el coseno de 90 vale cero).

Una vez calculada la semejanza contra cada documento de la colección y la consulta, el sistema es capaz de ordenar todos los documentos de la colección en orden decreciente de su grado de semejanza con la consulta, incorporando de este modo a los resultados aquellos documentos que satisfacen sólo parcialmente los términos de la consulta. Se efectúa, en consecuencia, equiparación parcial.

2.3 Cálculo de la semejanza entre documentos

En el modelo vectorial tanto un documento como una consulta se representan como un vector *t*-dimensional, siendo *t* el número de términos, donde cada valor asociado con un término representa su peso dentro del documento o consulta. Usando esta representación, es posible evaluar el grado de semejanza entre la consulta y los documentos como una función entre dos vectores.

La definición formal del modelo vectorial es la siguiente:

Sean *t* el número de términos en el sistema con el conjunto de términos $K = \{k_1,...,k_t\}$ y d_j un documento.

Un peso $w_{i,j} \geq 0$ es asociado con el par (k_i, d_j), con $w_{i,j} = 0$ cuando el término no aparece en el documento. De igual manera, $w_{i,j} \geq 0$ es el peso asociado con el par (k_i, q). Entonces, la consulta q está representada como el vector $q = (w_{1,q}, w_{2,q}, w_{3,q}, \dots, w_{t,q})$, y el vector de un documento d_j está representado por $d = (w_{1,j}, w_{2,j}, w_{3,j}, \dots, w_{t,j})$.

Dada esta representación, el grado de semejanza entre un documento d_j y la consulta q es, típicamente, cuantificado por el coseno del ángulo entre los vectores que los caracterizan. Esto es:

Cálculo de la semejanza.

$$sem(d_j q) = \frac{d_j \bullet q}{|d_j| \times |q|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}} \quad (2.15)$$

donde $|d_j|$ y $|q|$ son los módulos de los vectores del documento y de la consulta. Los pesos de los términos índices $(w_{i,j})$ se determinan en base a la frecuencia, en que los términos ocurren dentro de un documento y la frecuencia en que el término ocurre en todos los documentos.

Sea *N* el número total de documentos en el sistema, n_i el número de documentos en que el término k_i aparece, y $frec_{i,j}$ la frecuencia del término k_i en el documento d_j.

Si el término k_i no aparece en el documento d_j entonces $frec_{i,j} = 0$. Además, sea idf_i, la frecuencia inversa de documento para k_i, definida por

$$idf_i = \log \frac{N}{n_i} \quad (2.16)$$

el peso del término k_i en el documento d_j está dado por

$$w_{i,j} = frec_{i,j} \times idf_i \quad (2.17)$$

Esta estrategia para los pesos de los términos es llamada esquema $tf \cdot idf$.

2.4 Secuencias Frecuentes Maximales (SFM)

Una secuencia frecuente maximal, es una secuencia de palabras que debe aparecer en un número igual o superior al umbral dado (por ejemplo, documentos, oraciones, etc.) y además, no debe estar contenida en otra secuencia de palabras.

Las SFM se pueden usar para calcular la semejanza entre documentos, por ser un trabajo parecido a la construcción de un árbol de términos latentes, se menciona. Ambos se pueden usar para calcular la semejanza entre documentos, pero las SFM más largas tienen un peso mayor; en un árbol de términos latentes todos los términos latentes tienen el peso asignado por $tf \cdot idf$.

A continuación, se presenta la definición formal de Secuencias Frecuentes Maximales. Para entender esta es necesario asumir, que D es un conjunto de textos (por texto nos referimos a un documento completo o incluso a una sola oración), donde cada texto consiste en una secuencia de palabras [Ahonen-Myka H., 2002].

Definición 1. Una secuencia p = a1…ak, es una subsecuencia de una secuencia q si todos los elementos ai, $1 \leq i \leq k$, ocurren en q y además ocurren en el mismo orden que en p. Si una secuencia p es una subsecuencia de una secuencia q, entonces se dice que p ocurre en q.
Definición 2. Una secuencia p es frecuente en D si p es una subsecuencia de al menos σ textos de D, donde σ es un umbral de frecuencia predefinido.
Definición 3. Una secuencia p, es una secuencia frecuente maximal en D si no existe alguna otra secuencia p´ en D, tal que p es una sub secuencia de p´y p´ es frecuente en D.
Según [Kovács L. & Ahonen-Myka H.,2001] la razón para extraer SFM en lugar de secuencias de tamaño fijo, es porque las SFM tienen una representación flexible y compacta.

2.5 Estructura Lingüística

El siglo XX instaura lo que se denomina la lingüística moderna, relacional o estructural, cuyo fundador reconocido es el lingüista suizo, nacido en Ginebra, Ferdinand de Saussure (1857-1913).

Ferdinand de Saussure, considerado el padre de la lingüística, ha influido en las generaciones posteriores de una manera decisiva. Esta influencia fue ejercida a partir de una recopilación de sus conferencias, reconstruidas a partir de los cuadernos de apuntes de sus discípulos, que se publicó por primera vez en 1916.

La lingüística, dice Saussure, es una parte de una ciencia general más vasta, a la que se llama semiología y que estudia la vida de los signos en la vida social. Su fin es reconocer las reglas que gobiernan su generación, producción, transmisión etc. La lingüística es entonces, solo una parte de semiótica, ya que trata un tipo particular de signos.

2.5.1 Signo

Es un elemento que sirve para establecer comunicación entre un emisor y un receptor; es un elemento que sirve para indicar algo, es decir el signo forma parte de un código (una serie de convenciones preestablecidas, comunes tanto al emisor como al receptor).

Ante la presencia de signos nos encontramos con un proceso de significación:
Así cuando el campanero (emisor) hace sonar las campanas en la iglesia, las campanadas (signo) indican a la gente del pueblo (receptor) la celebración de misa en esta, es decir las campanadas en la iglesia significan la celebración de misa.

2.5.2 Signo lingüístico

Esta definición es la base sobre la cual se sostiene la semiología (Ferdinand de Saussure, Curso de Lingüística General, 1916). El signo lingüístico no vincula un nombre con un objeto, sino un concepto con una imagen acústica.

El enfoque de Saussure, sostiene que todas las palabras tienen un componente material (una imagen acústica), al que denominó significante y un componente mental referido a la idea o concepto representado por el significante, al que denominó significado.

Significante y significado conforman un signo

La lengua no es una nomenclatura, una lista de nombres que se refieren a cosas. Esto es un cambio fundamental en la lingüística del siglo XX llamada relacional. El signo es un mediador entre los hombres y las cosas, es a través de que aprendemos lo real, lo real se nos ofrece hecho lenguaje. (El lenguaje no refleja al mundo, sino que lo configura).

El signo lingüístico, es una entidad de dos caras, que une un concepto con una imagen acústica, un significado y un significante. El significante o imagen acústica es la parte material del signo, por ejemplo, para un mismo concepto, cada lengua tendrá un significante diferente: cat, gato. Los significantes funcionan dentro de un sistema solidario, relacional; así podemos diferenciar pato, gato, rato, mato, etc., porque en cada caso cambiamos algún fonema.

El significado, es el concepto acerca de una cosa. Si se dice árbol, la referencia no es a un árbol en particular, sino a una abstracción que hizo el hombre al nominar así a todas las cosas que vio que tenían un tronco y una copa. El concepto es una idea, pero no una idea dada de antemano o que pueda funcionar aisladamente.

2.6 Concepción del vocablo *término latente*

En este trabajo el vocablo *término latente*, es usado para darle una denominación al conjunto de colocaciones de palabras, que posiblemente denotan un signo lingüístico sin importar cuantas palabras la compongan.

“las colocaciones pueden ser vistas como una escala con grados diferentes de intensidad de relación, entre las palabras, que van desde las combinaciones idiomáticas hasta las combinaciones libres.

En un extremo de la escala hay combinaciones idiomáticas completas como, por ejemplo, estirar la pata, donde ni la palabra estirar, ni la pata pueden ser reemplazadas sin destruir el significado de la combinación. En este caso —y en todos los casos de combinaciones idiomáticas— el significado de toda combinación no está relacionado con los significados de sus componentes. Este tipo de combinaciones también pueden llamarse frasemas, como propone Mel’čuk (Mel’čuk, 1996).

En el otro extremo de la escala hay combinaciones totalmente libres de palabras, como, por ejemplo, ver un libro, donde cualquier palabra de la combinación puede ser sustituida por un conjunto grande de distintas palabras y el significado de toda la combinación es la suma de los significados de sus palabras constituyentes.” [Gelbukh-A.&Sidorov-G.,2006].

2.6.1 El concepto de Término para el Modelo Vectorial

En el Modelo Vectorial el vocablo Término significa: Cualquier conjunto de caracteres sin espacios intermedios y sin signos de puntuación. Existen tratamientos para cada uno de estos conjuntos de caracteres, que son útiles para ahorrar espacio cuando son almacenados, pero el significado de forma abstracta es el mismo.

2.6.2 El Término Latente un posible Signo Lingüístico

Las combinaciones diferentes de cadenas de caracteres que aparecen en una colección de documentos son los posibles términos, denotando a término como signo lingüístico no como cadena de caracteres.

El conjunto de combinaciones de cadenas que se genera de un documento, esta dada por la longitud del documento y el número de cadenas diferentes, que este documento contiene, de manera empírica, en este trabajo se estableció como umbral para las combinaciones de palabras un número de apariciones superior a uno.

Como se explicó, en las limitaciones de este trabajo, un “*Término latente*” es latente porque cabe la posibilidad, que a pesar de superar el umbral mínimo para ser tomado en cuenta, no tenga significado alguno, de ahí que sea un posible signo lingüístico y no un signo lingüístico.

Capítulo III
El modelo vectorial extendido

"La más desprevenida observación de nuestro comportamiento, de las condiciones de nuestra vida intelectual y social, de la vida de relación, de los nexos de producción y de intercambio, nos muestra que utilizamos a la vez, y a cada instante, varios sistemas de signos: los del lenguaje, los signos de la escritura, los "signos de cortesía", de reconocimiento, los signos reguladores de los movimientos de los vehículos, los "signos exteriores" que indican condiciones sociales, los "signos monetarios", los signos del arte en sus variedades (música, imágenes, reproducciones plásticas); en una palabra, y sin ir más allá de la verificación empírica, está claro que nuestra vida entera está presa en redes de signos que nos condicionan, al punto de que no podría suprimirse una sola sin poner en peligro el equilibrio de la sociedad y del individuo".

Émile Benveniste

Resumen

La diferencia entre el modelo clásico y esta extensión no está en el cómo calcular la semejanza entre documentos, está en la construcción del archivo inverso y en el uso de términos latentes seleccionados de una estructura de datos complementaria (árbol de términos latentes), para calcular la semejanza entre documentos.

Dada la semejanza entre el modelo vectorial y la presente extensión, este capítulo hace una descripción de las características que diferencian al modelo clásico de esta extensión. Estas diferencias están dadas en las estructuras de datos usadas para seleccionar los términos a usar, en el cálculo de la semejanza entre la consulta y la colección de documentos.

Este trabajo lleva por título "*Construcción de un árbol de términos latentes y su uso en el cálculo de la semejanza entre documentos*". ¿Cómo se calcula la semejanza entre documentos?, ¿Qué es un árbol de términos latentes? y ¿Cómo se construye un árbol de términos latentes?, son las preguntas a las que responde este capítulo.

3.1 Diccionario de términos y archivos inversos

Las distintas palabras que aparecen en la colección de documentos son para el modelo clásico vectorial un término [Salton 1989], la colección de todos los términos diferentes de una colección de documentos se llama diccionario de términos, el diccionario de términos es parte del archivo inverso. El archivo inverso es la estructura de datos que se consulta, para evitar la relectura de la colección de documentos, cuando se necesita información sobre ocurrencias o posiciones de un término en la colección de documentos.

Los pasos para la construcción de archivos inversos [Sidorov-G.,2011] son:

1) Reunir todos los documentos para su indización.
2) Partir los documentos en cadenas de caracteres para su posterior almacenamiento como términos en el archivo inverso.
3) Pre-procesamiento lingüístico de las cadenas obtenidas.
4)Indizado de los documentos por cada término que aparece en él.

El archivo inverso está compuesto por un diccionario de términos y una lista de identificadores, de documentos que son deducidos de una colección de documentos, para cada término en la colección. Esta idea es central para el concepto *índice invertido*, importante en la recuperación de información. El nombre es en realidad redundante: un índice siempre mapea detrás de los términos, las partes de un documento donde ocurre el término. Sin embargo, el índice invertido o archivo invertido, se ha convertido en el término estándar en recuperación de información. Se muestra la idea básica de un índice invertido en la Figura **3.1 Archivo inverso** siguiente extraída del libro [MAN,2009].

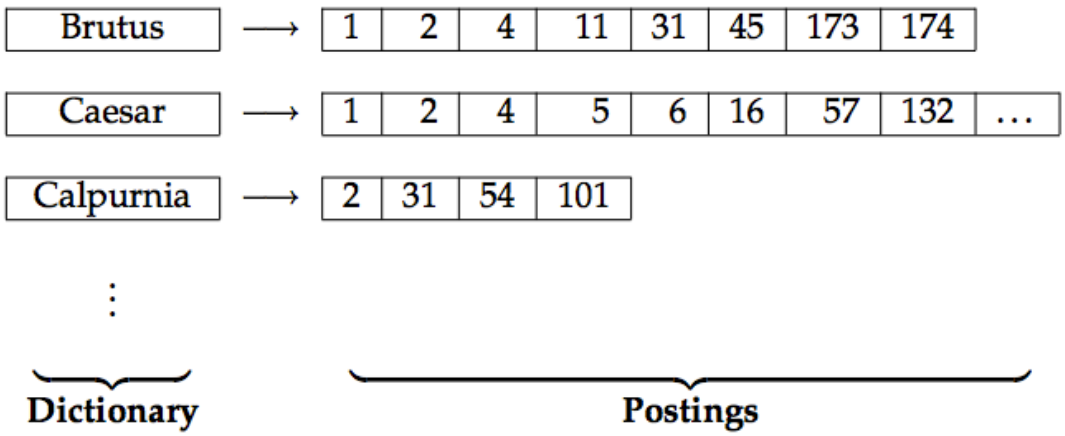


Figura 3.1 Archivo inverso

Por ejemplo para el siguiente par de documentos (**Figura 3.2**), primero se obtendrá el conjunto de términos que los integran, y en seguida se construye un archivo inverso en el que se guarda el término, el número de documentos en los que ocurre y los identificadores de esos documentos.

Documento Uno
La programación orientada a objetos es un paradigma relativamente nuevo

Documento Dos
Lo nuevo en las bases de datos es la programación de sus algoritmos

Término	**id de documento**		**Término**	**id de documento**
La	1		Lo	2
Programación	1		Nuevo	2
orientada	1		En	2
A	1		Las	2
Objetos	1		Bases	2
Es	1		De	2
Un	1		Datos	2
Paradigma	1		es	2
Relativamente	1		La	2
Nuevo	1		Programación	2
			Sus	2
			Algoritmos	2

Figura 3.2 Documentos Uno y Dos

Término	Número de documentos con el término	Lista de documentos
la	2	1, 2
programación	2	1, 2
orientada	1	1
a	1	1
objetos	1	1
es	2	1, 2
un	1	1
paradigma	1	1
relativamente	1	1
nuevo	2	1, 2
lo	1	2
en	1	2
las	1	2
bases	1	2
de	1	2
datos	1	2
sus	1	2
algoritmos	1	2

Figura 3.3 Archivo Inverso construido a partir de los documentos Uno y Dos

Cada documento es dividido en pares, identificador de término (IdTermino) e identificador de documento (IdDocumento), esto se puede ver en la **Figura 3.3,** esta operación se repite hasta que la colección de documentos se agota o el espacio en memoria reservado para almacenar estos pares se llena, en seguida este bloque es escrito en el archivo inverso. Cada identificador de Documento de la **Figura 3.4** hace referencia a una de dos opciones.

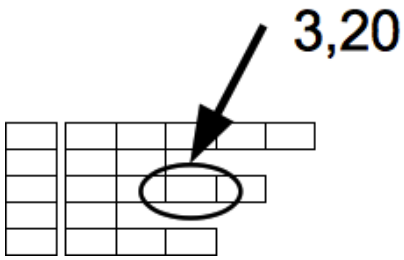


Figura 3.4 Diccionario de datos con sus correspondientes identificadores de documentos que lo contienen y la posición en que aparecen o el número de ocurrencias en ese documento.

Un conjunto de pares donde el primero de cada par es el índice del documento en que ocurre el término y el segundo, la posición de ese documento en que el término ocurre.

El archivo inverso compuesto por un diccionario e índices posicionales, también puede ser visto, como una estructura de datos tridimensional, donde cada identificador de documento referencia las posiciones dentro de ese documento, en que el término ocurre.

Esta estructura tridimensional, es una forma fácil de implementar un archivo inverso, está compuesta por un diccionario de términos y una lista de identificadores de documentos, a cada identificador de documentos se le asigna una lista de posiciones, en que el término ocurre en ese documento.

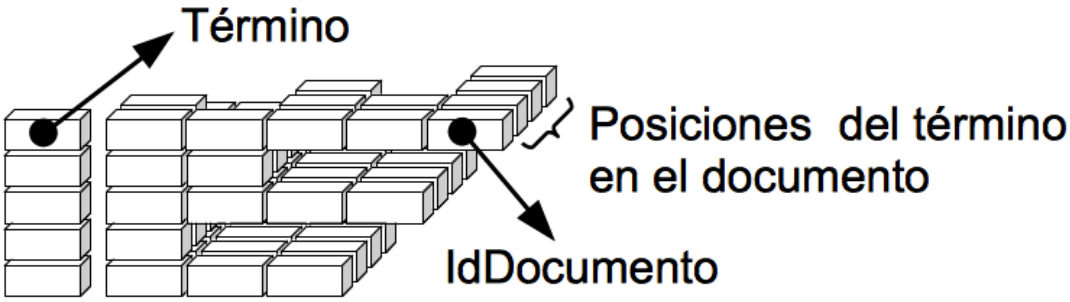


Figura 3.5 Diccionario de datos con sus correspondientes identificadores de documentos que lo contienen y la posición en que aparecen en una estructura tridimensional.

En la figura anterior (**Figura 3.5**) el diccionario de términos, es la columna señalada con la leyenda Término, los identificadores de los documentos que tienen al término, son los renglones correspondientes con cada elemento del diccionario, marcados en la figura como IdDocumento, las posiciones son la guardadas en el eje z del dibujo, la frecuencia del término en el documento, es igual al número de posiciones del término en el documento.

En el modelo clásico vectorial es indispensable tener una estructura de datos, dónde guardar los términos de la colección de los documentos y las posiciones en que los términos aparecen en los documentos, a esta estructura se le denomina archivo inverso.

3.2 Índices de pares de palabras

Un enfoque para manejar frases es considerar todos los pares consecutivos de términos en un documento como una frase. Por ejemplo, el texto Friends, Romans, Countrymen generaría los pares de palabras:
Friends romans
Romans contrymen

En el modelo vectorial, se puede tratar a cada uno de estos pares de palabras, como un término de diccionario. El uso de un diccionario de pares de palabras, expande el tamaño del vocabulario. Para tratar frases con longitud superior a dos, se pudiera proceder de la misma manera en el modelo clásico vectorial, teniendo como consecuencia un incremento en el tamaño del archivo inverso.

De construirse un archivo inverso de frases de cualquier longitud, el tamaño de este se incrementa hasta hacerse difícil de manejar, la propuesta que en este trabajo se da, es construir sólo fracciones de este archivo inverso; las útiles para calcular la semejanza de la consulta contra la colección de documentos.

La ventaja de tener un archivo inverso de términos con longitudes superiores a uno, es calcular la semejanza entre una consulta y la colección de documentos, tomando en cuenta a la consulta completa y no a los términos que la componen, dando un grupo menor de resultados, más exacto; en caso de no encontrarse la consulta en la colección de documentos, al construirse el árbol de términos latentes, se permite al usuario saber que términos latentes relacionados con lo que busca existen en la colección. El árbol de términos latentes además, permite encontrar términos latentes existentes en la colección con una longitud superior a la consulta, por ejemplo para la consulta *base de datos*, el árbol de términos latentes generado, tiene al nodo *base de datos de* y al nodo *de base de datos*, términos latentes que existen en la colección de documentos, pero que para descubrirlos sé construyo un árbol de términos latentes y no un archivo inversos de términos con longitudes superior a uno.

3.3 Delimitación de documentos y caracteres

3.3.1 Obtención de la secuencia de caracteres en un documento

Los documentos digitales que son la entrada a un proceso de indexación, suelen ser un conjunto de bytes en un archivo o en un servidor web. El primer paso del proceso consiste en convertir esta secuencia de bytes en una secuencia lineal de caracteres. Para el caso del texto en Inglés o español, esto es trivial , pero a menudo las cosas son más complejas. La secuencia de caracteres puede ser codificada por uno de varios esquemas, mono o poli byte. Para el caso del presente trabajo no hay problema, dado que los documentos son texto plano en español, con codificación UNICODE UTF-8.

3.3.2 Escoger la unidad documental

Escoger el documento o parte de él que será tomada en cuenta para agregarla en el archivo inverso, no es un problema en este caso de estudio pues los documentos son resúmenes de publicaciones de la revista computación y sistemas del Centro de Investigación en Computación del IPN, tomados de la base de datos *CyS.sql* anexa a este documento, facilitada por la biblioteca del Centro de Investigación en Computación del IPN. En casos tales que un documento es un libro completo, esto resultaría una mala idea, para estos casos existe la división de documentos en partes componentes, como título y resumen, que dependiendo del sistema de recuperación de información, en que se desarrolle y de las características de los documentos tratados, será adecuada al caso.

3.4 Armado del diccionario de términos

Determinar qué términos estarán contenidos en el diccionario, es el trabajo posterior a determinar la sección del documento, que será usada para componer el diccionario de términos. Esta tarea inicia con la separación del documento en cadenas (Tokenization).

3.4.1 División de los documentos en tokens

Dada una secuencia de caracteres y una unidad de documento definido, es la tokenización la tarea de cortar al documento en trozos, llamados tokens, tal vez, al mismo tiempo que se deshace de signos de puntuación.

A continuación un ejemplo:

Entrada: Amigos, romanos, compatriotas, me prestan sus oídos;
Salida: amigos compatriotas romanos me prestan sus oídos

Estos símbolos son a menudo conocidos de forma indistinta como términos o palabras, pero a veces es importante hacer una distinción simbólica entre token y tipo.

Token: es una instancia de una secuencia de caracteres en algún documento en particular, que están agrupados como una unidad semántica [MAN, 2009] útil para su procesamiento.

Tipo: es la clase de todos los tokens que tienen la misma secuencia de caracteres.

Lo que se indexa en el archivo inverso, es en primera instancia un Tipo dado, que en la práctica puede no ser idéntico a lo encontrado en el texto original (debido a la eliminación de signos de puntuación, agrupamiento, etc).

Cada lenguaje [MAN, 2009] presenta algunos temas nuevos. Por ejemplo, Francia tiene una variante de uso del apóstrofe, para un reducido artículo definido "la" antes de una palabra comenzando con una vocal (por ejemplo, l'ensemble) y tiene algunos usos del guión con los pronombres, pospuestos en los imperativos y las preguntas (por ejemplo, donnemoi 'Dame').

En otros idiomas el problema es más difícil. En alemán, se escriben nombres compuestos sin espacios (por ejemplo, 'lingüística computacional' Computerlinguistik, o “seguro de vida de empleado” Lebensversicherungsgesellschaftsangestellter).

Sistemas de recuperación de información para el Alemán, se benefician enormemente de la utilización de un módulo compuesto para dividir palabras compuestas, que suele ser implementada por ver, si una palabra se puede subdividir en múltiples palabras que aparezcan en el diccionario. Este fenómeno alcanza su límite con el caso de los principales idiomas de Asia oriental (por ejemplo, chino, Japonés, coreano y tailandés), donde el texto está escrito sin espacios entre las palabras.

Lo mencionado antes sobre la forma de obtener los términos que formaran el diccionario, éste a su vez parte del archivo inverso sirve para entender el porque éste trabajo funciona sólo para lenguajes como el español o el inglés donde no se da la fusión de palabras.

3.4.2 Eliminación de los términos más comunes: las palabras vacías

A veces, algunas palabras son tan comunes que parecen ser de poco valor para ayudar a seleccionar los documentos, que coincidan con una necesidad expresada en el

documento usado, para comparar con el resto de la colección. Estas palabras se llaman las palabras vacías.

En el capítulo dos ya se dijo que el modelo vectorial, es un modelo de bolsa de palabras, es decir, es un modelo al que no le importa el orden en que los términos aparezcan [MAN, 2009], si se toma en cuenta está limitación pareciera no tener diferencia el que se usaran o no palabras vacías, para calcular la semejanza entre documentos. Para está extensión el tomar en cuenta a las palabras vacías, es de gran importancia, debido a la necesidad de tener a estas, en el archivo inverso para su uso en la construcción de un árbol de términos latentes, que permita solventar la limitación del modelo vectorial como un modelo de bolsa de palabras, para convertirlo en un modelo en el que se puedan usar términos latentes, para calcular la semejanza entre documentos. Debido a lo anterior, el presente trabajo toma en cuenta, todos los términos encontrados en la colección de documentos, para construir el archivo inverso (situación diferente de usarlos para calcular la semejanza entre documentos, sin que así lo dicte la voluntad del usuario).

3.4.3 Normalización (clases equivalentes de términos)

Después de haber dividido los documentos (y también la consulta) en tokens, el caso en que los tokens en la consulta se encuentren en el diccionario de términos, no tiene complicación. Sin embargo, hay muchos casos en que dos secuencias de caracteres son distintas que debieran ser tomadas como iguales. Por ejemplo, si se busca USA, se espera tomar en cuenta documentos que contengan U.S.A..

Normalización simbólica, es el proceso en el que varios tokens son tomados como iguales, en el presente trabajo se toman como iguales tokens, sin importar que sus caracteres sean mayúsculos o minúsculos, pero no se tiene ninguna relación de equivalencia entre tokens diferentes.

3.4.4 Acentos, diacríticos y capitalización

Los acentos hacen diferenciar entre palabras por ejemplo numero se refiere a coordinar un conjunto cualquiera con el conjunto de los números, y número se refiere de manera abstracta a un elemento del conjunto de los números. En este trabajo no se eliminan los acentos debido a su utilidad, invirtiendo espacio extra para los términos latentes diferentes por causa de palabras con acentos.

La palabra Peña, puede referir a una roca grande o al apellido de una persona, igual que lo puede hacer la palabra peña. En este trabajo se transforman todos los tokens en palabras idénticas, pero con todos sus caracteres representados en minúsculas.

La palabra pena significa pesar o castigo, el significado de peña es totalmente distinto. En este trabajo los diacríticos igual que los acentos no son eliminados.

3.4.5 Stemming y lematización

Por razones gramaticales, los documentos usan diferentes formas de una palabra como: organiza, organizar y organización. Adicionalmente existen familias de palabras relacionadas por derivación con significados semejantes, tales como democracia, democrático y democratización. En muchas situaciones, es útil tener como resultado en una búsqueda documentos que contengan alguna palabra relacionada, como en los ejemplos anteriores, es decir si se escribe demócrata, se tome en cuenta a los documentos que contengan democratización.

La finalidad del stemming y la lematización, es reducir las formas inflexivas y algunas veces las formas derivadas de una palabra con una base común. Por ejemplo:
Am, are, is > be
Car, cars, car´s, cars´ > car

En el ejemplo anterior > significa produce. Sin embargo, las dos palabras difieren en significado. Stemming, se refiere a un proceso heurístico que trunca la terminación de las palabras. La lematización, se refiere al análisis morfológico de las palabras y a su uso correcto en el vocabulario, retornando la forma básica de la palabra o la forma guardada en el diccionario.

Si se encuentra la palabra saw, el stemming retorna s, mientras la lematización retornara see o saw, dependiendo si el token se uso como verbo o sustantivo.

El algoritmo más común para hacer stemming en inglés y además el que ha mostrado empíricamente ser muy efectivo, es el algoritmo de Porter [Porter 1980]. El algoritmo de Porter, consiste en 5 faces de reducción de palabras, aplicadas secuencialmente.

Experimentos y discusiones sobre lo positivo y negativo en cuanto al impacto del stemming y la lematización en ingles, se da en los trabajos de [Salton 1989], [Harman 1991], [Krovetz 1995], [Hull 1996]. Para éste trabajo, no se utilizó, ni stemming, ni lematización puesto que no son necesarios para mostrar la extensión del modelo vectorial a la que éste trabajo se refiere.

La finalidad de la presente extensión al modelo clásico vectorial no es reducir el tamaño del diccionario de términos generado por la colección de documentos o manipularlo para obtener falsos positivos, en caso de no encontrar al término buscado; utilidades que tienen la lematizacion y el stemming, por tal razón los términos encontrados en la colección de documentos, no son tratados con estas técnicas.

3.6 ¿Qué es un término latente para esta extensión del Modelo Vectorial?

La terminología para un tópico específico está dada igual que el significado de un signo, en un lenguaje dado, por un conjunto de acuerdos preestablecidos por quien domina el tópico. Un término latente para esta extensión del modelo vectorial, es una o más cadenas de caracteres, que de forma latente enlazan una idea o concepto con ella, obteniendo en caso de acierto un signo lingüístico, esta es la diferencia sutil entre el modelo vectorial y su presente extensión.

Los términos latentes, se usan en esta extensión al modelo vectorial para calcular la semejanza entre documentos y hacen la diferencia única entre está extensión y el modelo clásico.

3.7 Archivo inverso de términos latentes

Para poder comparar la exactitud del modelo clásico contra la extensión, se necesita una referencia fiable contra la cual comparar los resultados arrojados por ambos modelos, está referencia es el archivo de términos latentes de cualquier longitud, el archivo de términos latentes de cualquier longitud, se anexa en el documento *TerminosYUbicacionesEnDocumentos.txt*. En el que se encuentra para cada renglón, de izquierda a derecha, el término latente y en seguida una lista de documentos que contienen al término latente, lo que esta entre [], son las posiciones en el documento que precede a esta agrupación.

La longitud máxima de los términos latentes encontrados en la colección es dieciocho, por tal razón, el término latente más largo encontrado en *TerminosYUbicacionesEnDocumentos.txt*, tiene longitud dieciocho.

En la **Figura 3.6** se muestra el algoritmo utilizado para la construcción del archivo inverso de términos latentes de cualquier longitud. Nótese que cualquier longitud significa: Términos latentes de cualquier longitud, que existan en la colección de documentos.

- *Tmp=0*
- *L = máxima longitud del término*
- *Mientras Tmp < L*
- *Para todos los términos de longitud L*
- *Si existe una posición consecutiva a la posición del término actual*
- *Concatenar el término actual con el contenido en la posición consecutiva*
- *Guardar (este nuevo término, su*
- *longitud y posición)*
- *Tmp = L*
- *L = máxima longitud de todos los términos*
-
-
- *Guardar(término, longitud, posición)*
- *Si el término existe*
- *Actualizar el número de ocurrencias*
- *Guardar la posición y documento en que aparece*
- *Si el término es nuevo*
- *Guardar el nuevo término, su longitud e iniciar ocurrencias en 1*
- *Guardar la posición y documento en que aparece*

Figura 3.6 Seudocódigo para crear el archivo inverso de los términos latentes de cualquier longitud.

3.8 Descripción de un árbol de términos latentes

Un árbol de términos latentes, es un conjunto de niveles, los niveles tienen como elementos a nodos que representan términos latentes de igual longitud, como atributo cada nodo tiene una cadena que empata con el término latente que representa, y un archivo inverso. El archivo inverso de cada nodo dice qué documentos son los que tienen al término latente representado y las posiciones en que el término latente ocurre en los documentos que lo contienen.

La solución propuesta en el caso de esta extensión es construir solo los fragmentos útiles de este archivo inverso de términos latentes de cualquier longitud, existentes en la colección, para una consulta dada. Para construir un árbol de términos latentes (ATL_C) se necesita un Archivo Inverso (AI), construido a partir de la colección de documentos y una Consulta (C), los renglones de AI que corresponden con los términos de la consulta son los renglones que servirán para construir los fragmentos del archivo inverso de cualquier longitud, relacionados con la consulta, es decir los fragmentos útiles del archivo inverso de términos latentes de cualquier longitud.

Los nodos del ATL_C, tiene un Archivo Inverso para el término latente que representan, es decir los nodos del ATL_C representan sólo términos latentes existentes en la colección. Los nodos que representan términos latentes con longitud superior a uno, se generan a partir de los nodos del primer nivel del árbol, como se muestra en la **Figura 3.12** y del ATL_C se seleccionarán los nodos que representen a los términos latentes, que serán usados para calcular la semejanza entre documentos.

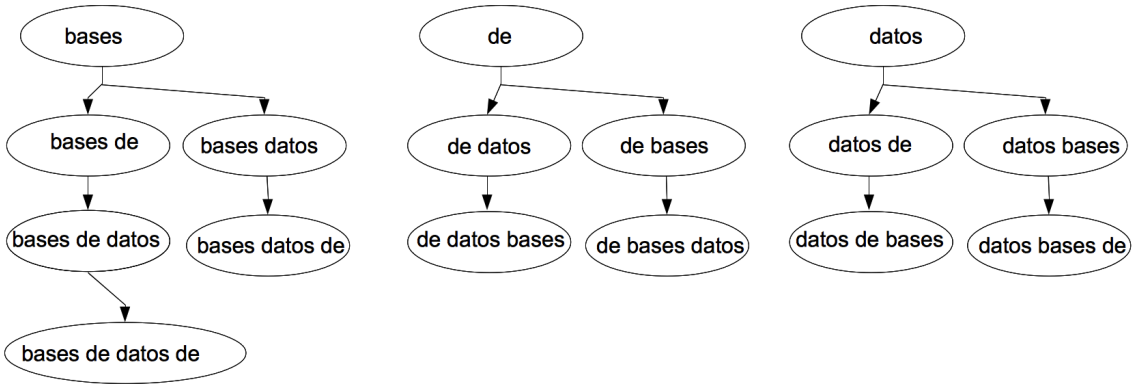


Figura 3.7 $ATL_{bases\ de\ datos}$

Los nodos del árbol de términos latentes que se puede ver en la **Figura 3.7,** sólo representan a términos latentes existentes en la colección de documentos, sin importar que la longitud de los términos latentes sea superior a la longitud de la consulta. Cada nodo del árbol tiene un archivo inverso con términos latentes de longitud igual a la profundidad en que se encuentra el nodo.

El árbol de términos latentes existentes en una colección de documentos ATL_C, que se genera para resolver una consulta, tiene como profundidad la longitud del término latente más largo encontrado en la colección de documentos, que involucra a un

subconjunto de los términos encontrados en la consulta, éste árbol de términos latentes es el que se implemento en éste trabajo.

El árbol de términos latentes, es un conjunto de nodos distribuidos en niveles, donde cada nivel contiene un conjunto de nodos que representan términos latentes con una longitud determinada.

3.9 Construcción de un árbol de términos latentes

Cada nodo del árbol de términos latentes tiene un archivo inverso correspondiente con los documentos y ocurrencias en estos documentos del término latente al que corresponden. El primer nivel del árbol se construye con el renglón del archivo inverso, que corresponde al término que el nodo representa, esto se puede ver en la **Figura 3.8**.

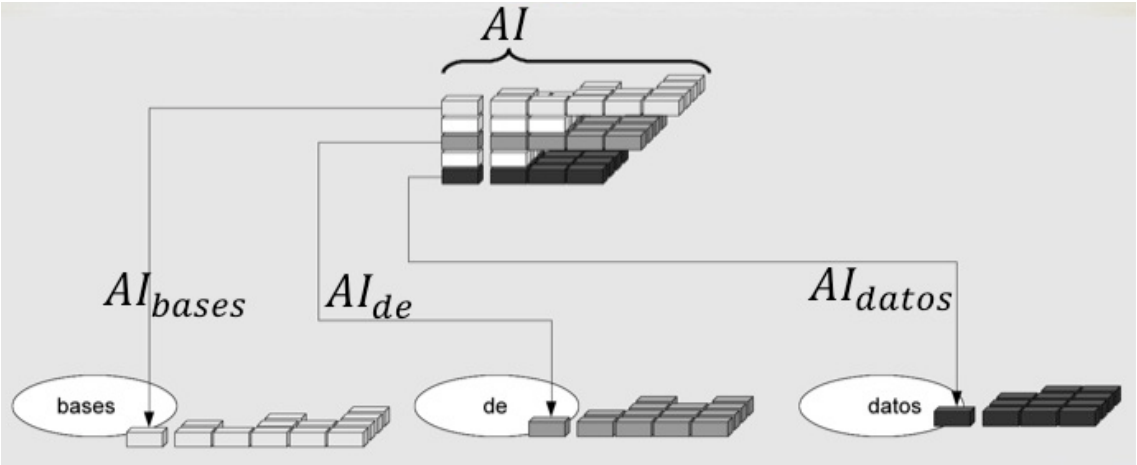


Figura 3.8 Archivo inverso que contiene los términos *bases, de* y *datos*

El segundo nivel del árbol es un conjunto de nodos, que se construye buscando en los nodos del primer nivel, documentos comunes con índices consecutivos, de encontrarse estos, se incluye en el árbol un nuevo nodo al que se le agrega un nuevo índice de documento y a éste se le asigna la posición o posiciones donde el término latente aparece en el documento. Es decir sólo se generan nuevos nodos que representan términos latentes existentes en la colección, en la **Figura 3.9** se puede observar la construcción de los nodos de nivel dos, a partir de los nodos de nivel uno.

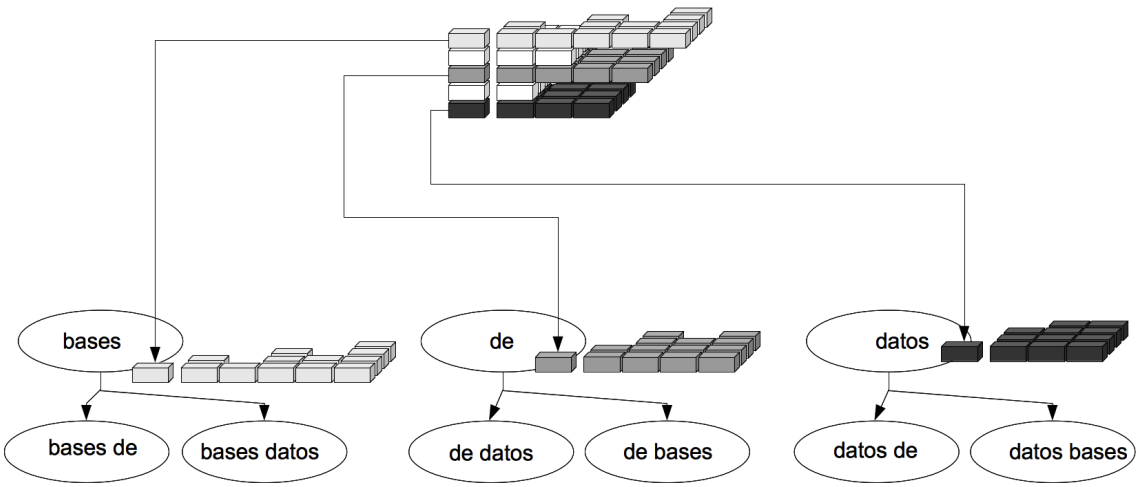


Figura 3.9 Fragmentos del Archivo inverso que se usan para crear el árbol de términos latentes de la consulta *bases de datos*

El resto de los niveles que se generan en un árbol de términos latentes, es el resultado de buscar posiciones consecutivas en documentos comunes de los nodos del último nivel existente con los nodos del primer nivel, esto da como resultado un árbol de términos latentes en los que cada nodo tiene como atributo un archivo inverso con un solo renglón, el renglón que tiene a los identificadores de los documentos en que ocurre el término latente y las posiciones en que ocurrió para ese documento, el árbol resultante para la consulta *bases de datos*, se ve representado en la **Figura 3.10**.

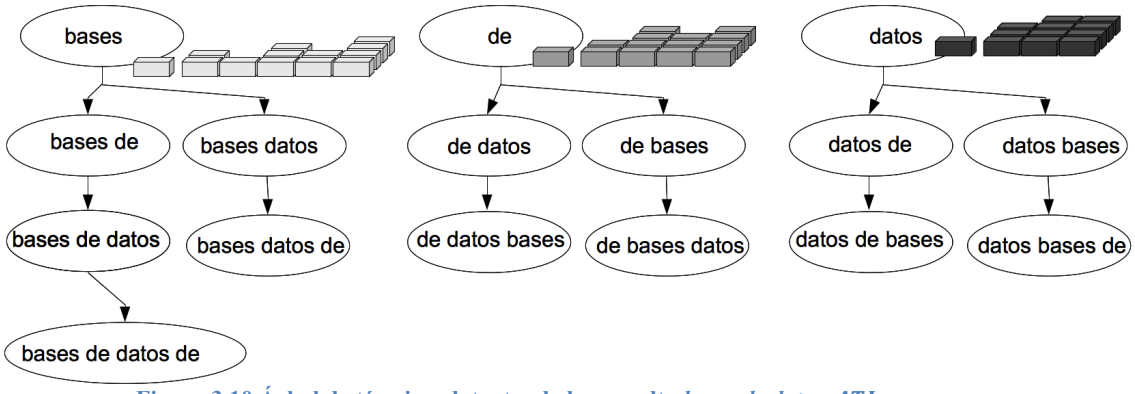


Figura 3.10 Árbol de términos latentes de la consulta *bases de datos* $ATL_{bases\ de\ datos}$

3.10 El esquema $tf \cdot idf$ en el modelo vectorial extendido

En el modelo clásico vectorial como ya se dijo en el apartado 2.3 t es el número de términos en el sistema y el conjunto de términos es $K = \{k_1, \dots, k_t\}$, donde d_j es un documento y D es una colección de documentos y $d_j \in D$. De manera implícita esta dicho que cada elemento de K esta compuesto por una sola palabra, la longitud de todos los elementos de K es uno, además

$C = \{t, \dots, t_w\}$, donde C es una consulta compuesta por w términos t. (3.1)

En el modelo vectorial extendido, los términos latentes tienen longitudes que van desde uno hasta la longitud de la consulta C, usada para comparar contra cada d_j de D. Para

representar esto se dice que K_1, es la colección de todos los términos latentes con longitud 1, encontrados en la consulta C, K_n es la colección de todos los términos latentes con longitud n encontrados en la consulta, y

$$\mathbb{K} = \{K_1 \cup ... \cup K_n\}. \quad (3.2)$$

$\mathbb{K}$ representa todos los términos latentes posibles basándose en una C. Esto es todas las cadenas generadas al permutar las palabras que la componen y las cadenas truncadas de estas permutaciones.

Los términos latentes existentes, son un subconjunto de $\mathbb{K}$ y se denotan por $\mathbb{K}_e$, es decir $\mathbb{K}_e \subseteq \mathbb{K}$; que un término latente sea posible no significa que tenga ocurrencias en la colección de documentos, $\mathbb{K}_e$ es el conjunto de términos latentes que además de ser posibles tienen ocurrencia en la colección de documentos.

En el modelo vectorial todos los términos presentes en C son usados para calcular la semejanza contra cada d_j de D, lo que da como resultado un subconjunto de D, en la presente extensión al modelo vectorial se toma un subconjunto de $\mathbb{K}_e$, para calcular la semejanza contra los documentos, donde el usuario que puede ser una persona u otro sistema selecciona un subconjunto de los términos latentes existentes ($\mathbb{K}_e$), denominado términos latentes seleccionados $\mathbb{K}_s$,al resultado de esta selección.

$$\mathbb{K}_s \subseteq \mathbb{K}_e \subset \mathbb{K} \quad (3.3)$$

En el modelo vectorial t es el número de términos en el sistema, en esta extensión $\mathbb{t}$ es el número de términos latentes posibles generados a partir de la consulta, $\mathbb{t}_e$ es el número de términos latentes existentes en la colección de documentos, $\mathbb{t}_s$ es el número de términos latentes seleccionados para calcular la semejanza de la consulta contra cada documento de la colección.

El grado de semejanza entre la consulta contra cada documento se calcula así:

$$sem(d_j C) = \frac{d_j \bullet C}{|d_j| \times |C|} = \frac{\sum_{i=1}^{\mathbb{t}_s} w_{i,j} \times w_{i,C}}{\sqrt{\sum_{i=1}^{\mathbb{t}_s} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{\mathbb{t}_s} w_{i,C}^2}} \quad (3.4)$$

Por ejemplo para $C =$ *base de datos* existen tres términos con longitud 1, seis términos latentes con longitud 2 y seis términos con longitud 3, listados a continuación:

base
de
datos

base de
base datos
de datos
de base

datos de
de datos

base de datos
base datos de
de datos base
de base datos
datos de base
datos de base

además existe un número infinito de términos latentes posibles, basados en la combinación de los w, términos que componen a la consulta por ejemplo:

de de
base de datos de
datos de base de datos

todos estos términos latentes posibles corresponden con $\mathbb{K}$. Para la colección de documentos de la revista Computación y Sistemas $\mathbb{K}_e$ corresponde con:

de
base
datos

de de
de base
de datos
base de
datos de

de base de
de datos de
base de datos
de base de datos
base de datos de

Como se puede observar $\mathbb{K}_e \subset \mathbb{K}$ y de $\mathbb{K}_e$ se elegirá a los términos que serán usados para calcular la semejanza entre C y cada d_j de D, es decir $\mathbb{K}_s \subseteq \mathbb{K}_e$, que un término latente sea posible, no significa que tenga ocurrencias en la colección de documentos, la implementación que para este trabajo se hizo, solo genera nodos que representan términos latentes existentes ($\mathbb{K}_e$).

Este trabajo no tiene como finalidad sustituir a Google, la finalidad de este trabajo es extender al modelo vectorial, para hacerlo más exacto en la comparación de documentos, las SFM encuentran secuencias maximales y después las usan para calcular semejanzas entre documentos; la presente extensión también descubre secuencias maximales y las pone a disposición del usuario, para que decida usarlas o no en el cálculo de la semejanza. En caso que la consulta no se encuentre idéntica en la colección de documentos, el usuario tiene la opción de seleccionar consultas alternativas.

Capítulo IV
Implementación

No es posible confiar en un código que no haya usted creado por sí mismo. (En especial códigos que provengan de empresas que emplean a personas como yo).

Ken Thompson

Resumen

Una forma tangible para probar la eficiencia de la extensión al modelo vectorial, se puede hacer manifiesta al evaluar una implementación de ésta, contra la implementación del modelo clásico, en este capítulo se describen las partes y funcionamiento de la implementación que se hizo de está extensión al modelo vectorial.

4.1 Características de la fuente de documentos

La cantidad de documentos usados es de 267, correspondientes con igual número de resúmenes de las publicaciones de la revista Computación y Sistemas en el período 1997-2009. La base de datos de la que se extrajeron estos resúmenes es *CyS*, específicamente del campo RESUMEN de la tabla títulos.

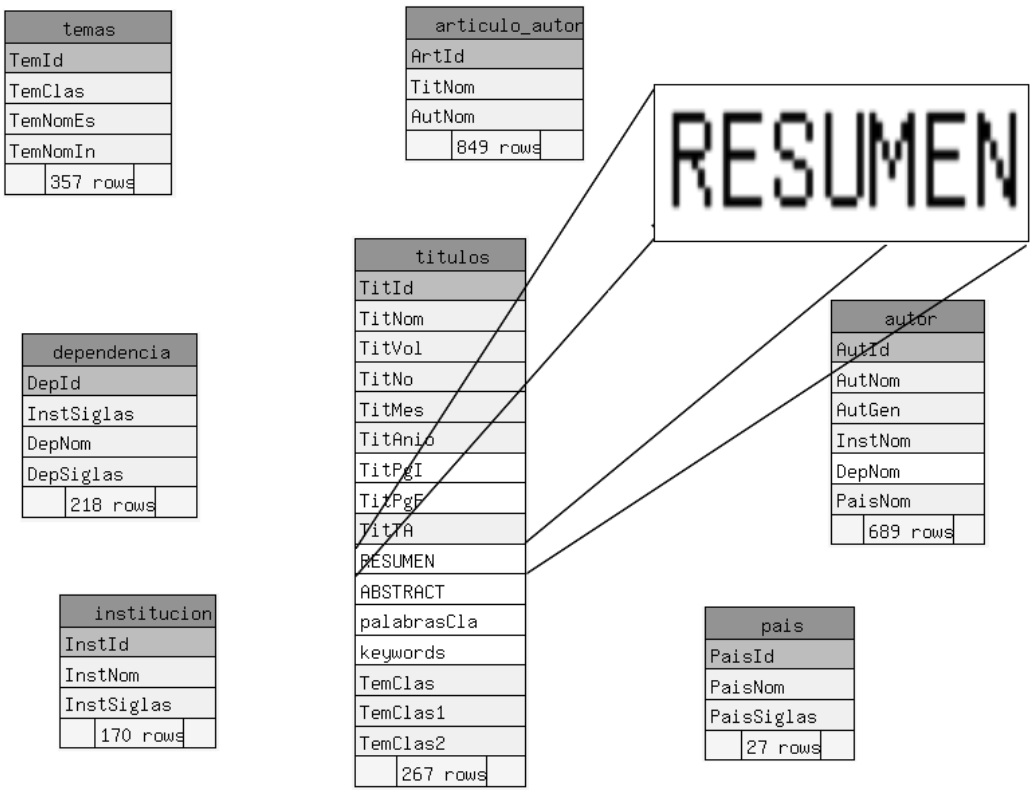


Figura 4.1 Campo RESUMEN de la tabla títulos

Este conjunto de documentos está guardado en el archivo de texto plano Documentos.txt, que se presenta como anexo de este trabajo, igual que la base de datos guardada con el nombre *CyS.sql*; en la **Figura 4.1** se ve el conjunto de tablas que forman la base de datos, también se resalta el campo resumen de la tabla títulos, que es el campo dónde se obtienen los documentos de la colección.

4.2 Partes de la implementación

La implementación puede ser descrita mediante una abstracción, que consta de seis pasos generales, estos pasos y el orden en que se ejecutan están listados en el siguiente gráfico. Concretamente cada uno de estos pasos lleva a cabo un conjunto de tareas que son descritas en éste capítulo, esto se ve representado en la **Figura 4.2**.

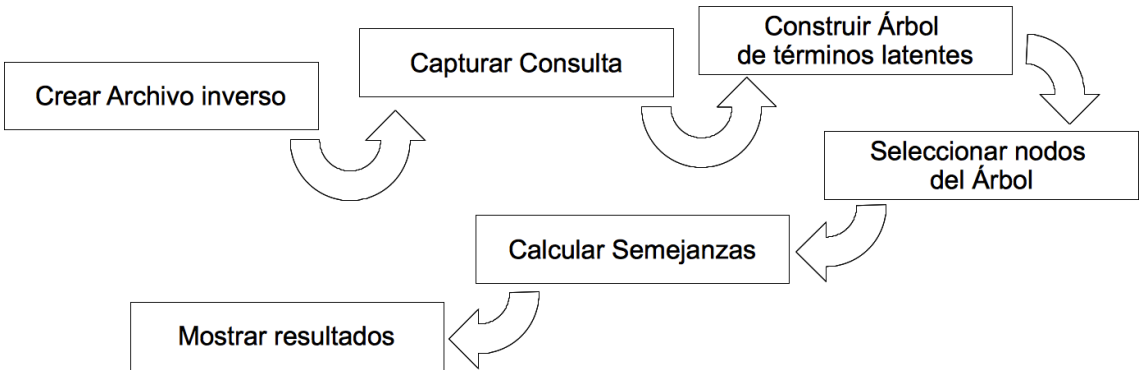


Figura 4. 2 Seis partes de la implementación

La primer tarea del sistema, es crear el archivo inverso de los términos encontrados en la colección de documentos, está tarea da inicio cuando en el método *main* de la clase *Main* se instancia un objeto de tipo *CrearPosting*, que a su vez instancía un objeto de tipo *OperacionesConPostings*, está instancia recupera los documentos de la base de datos y pide al usuario la consulta a utilizar para calcular la semejanza entre la consulta y cada documento.

Una vez que la instancia de clase de tipo *CrearPosting*, tiene la colección de documentos guardada como un conjunto de cadenas de caracteres, guardadas en una lista; procede a partir cada cadena en términos tomando como limite entre cada uno, un espacio en blanco, para cada uno de estos nuevos términos invoca al método insertar que se encarga de instanciar objetos de tipo *Renglon,* en caso de no encontrar una instancia que represente al término actual, en caso de existir un objeto de tipo *Renglon* que represente al término actual se busca en el miembro *idsDocumentos* de *Renglon*, un objeto de tipo *IDDocumento* que represente al documento correcto, de no encontrarse se instancia uno nuevo y por último en el objeto de tipo *IDDocumento* correcto se agrega a su miembro *indicesDelTermino,* la posición correspondiente al término actual.

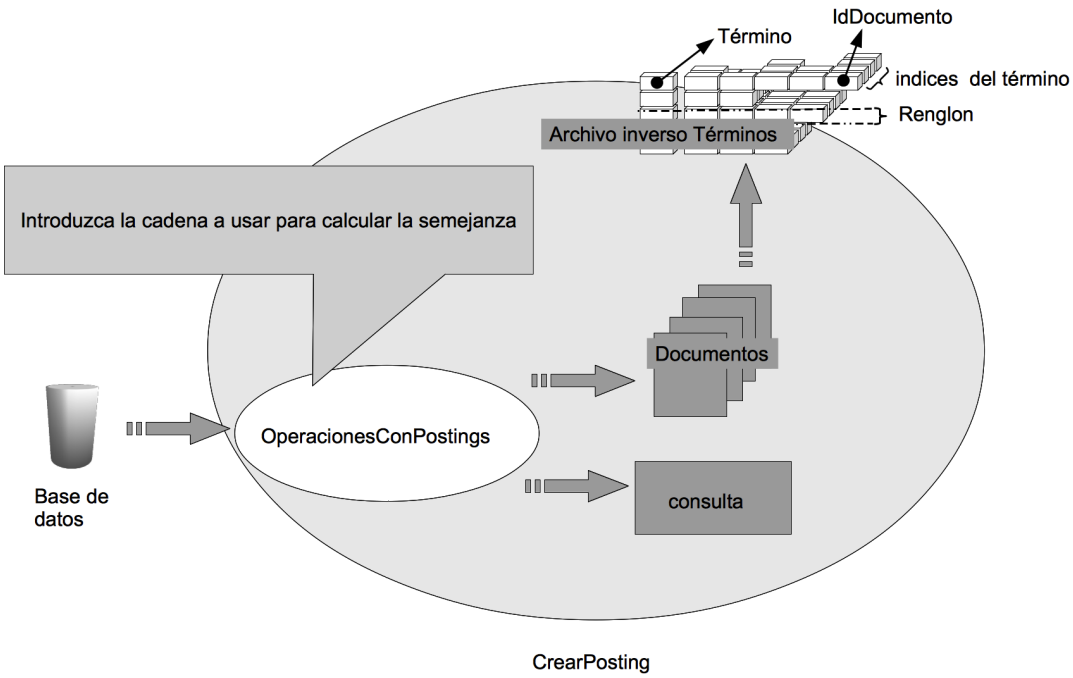


Figura 4. 3 Crear Archivo inverso y capturar consulta

En la ilustración anterior (**Figura 4.3**) se resume lo antes explicado; en una instancia de la clase *OperacionesConPostings,* que es un argumento para el constructor de la clase *CrearPosting*, se crea el archivo inverso de términos para la colección de documentos y se pide la consulta a comparar, contra la colección de documentos al usuario. El archivo inverso de términos ahora es un miembro de la instancia de clase de tipo *CrearPosting*.

Para continuar con el siguiente par de tareas: construir árbol de términos latentes y seleccionar nodos del árbol, debe observarse que el constructor de la clase *CalcularSemejanzas*, recibe como argumento una instancia de la clase *SeleccionarNodos* y el constructor de esta a un objeto de tipo *ConstruirATL*. Para explicar esta parte de la implementación se iniciará por la clase, que primero se instancía *ConstruirATL*.

El miembro *Query* de la clase *CrearPosting,* tiene ámbito de clase, lo que permite a todas las instancias de clase y a la clase misma acceder a este miembro, además la clase a la que pertenece no tiene modificador de acceso, lo que la hace una clase visible para todas las clases del mismo paquete. Esta característica del miembro *Query* de la clase *CrearPosting,* permite instanciar un objeto de tipo *ConstruirATL,* que recibe una referencia al objeto *Query* y otra al archivo inverso de términos en el método *main* de la clase *Main*.

Una vez instanciado un objeto de tipo *ConstruirATL*, este construye un árbol de términos latentes que es útil como argumento para el constructor de la clase *SeleccionarNodos*, que despliega para el usuario un catálogo de nodos en el árbol de términos latentes, para que elija de ellos los que se usaran para calcular la semejanza entre documentos, captura una cadena que el usuario introduce para seleccionar los nodos que se usaran, para calcular la semejanza y crea un arreglo que contiene a los nodos seleccionados del árbol de términos latentes.

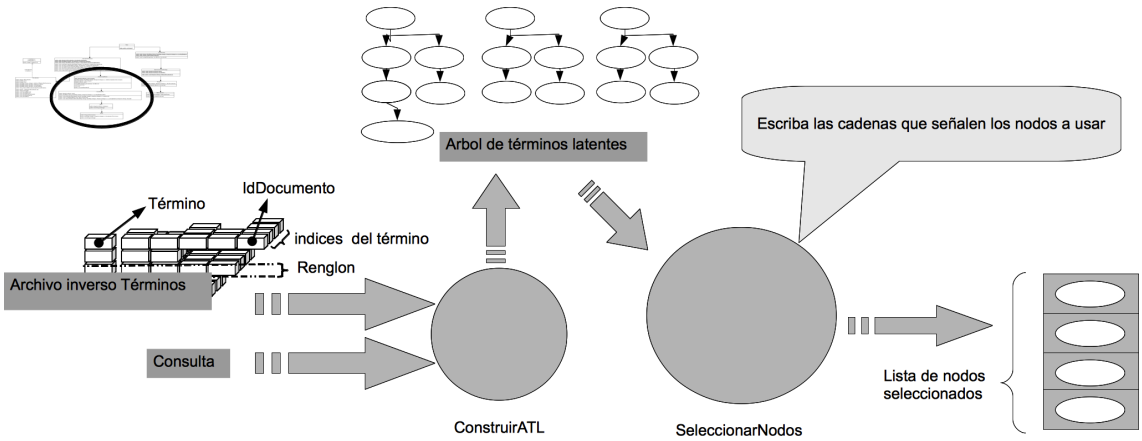


Figura 4. 4 Construir Árbol de términos latentes y seleccionar nodos

La construcción del ATL_C requiere del AI y de la C, esto se representa en la **Figura 4.4** la clase encargada de construir el árbol de términos latentes es ConstruirATL, una vez construido el árbol de términos latentes, su atributo árbol puede ser usado como argumento para instanciar un objeto de tipo *SeleccionarNodos*, que después de mostrar un catálogo con los nodos existentes en el árbol, recibe del usuario una cadena que señala los nodos a usar en el cálculo de la semejanza, entonces el atributo *terminosLatentesSeleccionados* de la clase *SeleccionarNodos,* se puede usar como argumento para el constructor de la clase *CalcularSemejanzas*.

Calcular semejanzas y mostrar resultados son las tareas de la implementación que en seguida serán explicadas, *CalcularSemejanzas,* es una clase que necesita como argumento a la lista de nodos seleccionados del árbol de términos latentes, para calcular la semejanza entre los documentos de la colección y la consulta, la clase *Documento* es una abstracción de los documentos involucrados en la selección de nodos del árbol de términos latentes, esta clase implementa la interface *Comparable* para poder ordenar de manera decreciente a los documentos, tomando en cuenta su grado de semejanza con la consulta dada.

El resultado que se muestra al usuario, es un conjunto de documentos, los cuales están ordenados, siendo el primero el que es más semejante a la consulta y el último el menos semejante, para hacer más fácil la explicación de la implementación, se escribió la clase *Documento*, las instancias de ésta clase sólo referencian a los pesos ponderados de los términos latentes, que son representados por los nodos seleccionados del árbol de términos latentes.

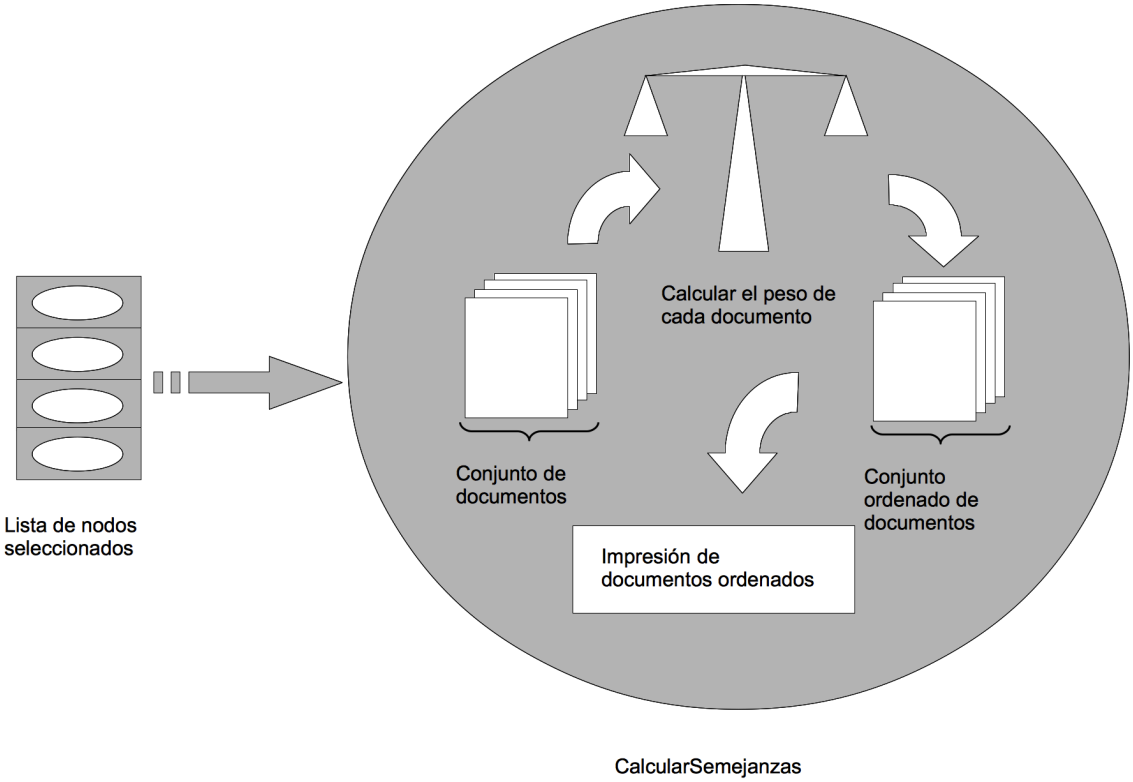


Figura 4. 5 Calcular semejanzas y mostrar resultados

La figura anterior (**Figura 4.5**) explica el último par de tareas que la implementación ejecuta, las instancias de la clase Documento, tienen el atributo *terminosYNumeroDeOcurrencias*, que es una tabla hash, está tabla tiene como llave a un término latente, y como valor el número de ocurrencias de éste término latente en el documento que la instancia representa, claro esta que los términos latentes encontrados en éste atributo, son un subconjunto de los términos latentes representados por los nodos seleccionados del árbol de términos latentes.

4.4 Funcionamiento de la implementación

La aplicación inicia su ejecución escribiendo en la línea de comandos *java 14deEnero.java*, esto después de ubicarse en la dirección donde el archivo *14deEnero.class* esté guardado, ambos archivos anexos a este documento. Una vez que la aplicación se encuentra en ejecución, se debe introducir la consulta que será usada para construir un árbol de términos latentes existentes. Una vez construido este árbol, sus nodos son mostrados al usuario, para que seleccione los que quiera usar para calcular la semejanza contra la colección de documentos. Para hacer más fácil la explicación, a continuación se muestra un ejemplo.

En la **Figura 4.6 Introducción de consulta,** se observa que el programa pide al usuario que teclee la consulta a partir de la que se construirá el árbol de términos latentes. En seguida se introduce la consulta a usar, para construir un árbol de términos latentes $ATL_{consulta}$ a partir de la consulta y el Archivo inverso de términos latentes de longitud uno AI.

```
Introduzca la consulta a usar para construir el árbol de términos latentes
```

Figura 4. 6 Introducción de consulta

Se oprime la tecla retro y como se puede ver en la **Figura 4.27 términos latentes existentes para la colección de documentos que tienen a un elemento de $\mathbb{K}_e$,** se despliega en pantalla una lista de términos latentes existentes en la colección de documentos a partir de la consulta dada. Es decir se enumeran los términos latentes existentes para la colección de documentos, que tienen a un elemento de $\mathbb{K}_e$.

```
        base de datos
nivel del arbol = 1
de{1=[10, 13, 24, 38, 40, 48], 2=[5, 12, 17, 26, 34, 36, 69, 89, 108, 117, 133, 140, 142,
base{1=[19], 207=[31], 206=[99, 121], 138=[67], 201=[12, 75], 5=[93, 237], 6=[77], 9=[7],
datos{1=[39, 49], 207=[91], 206=[101], 201=[14], 200=[44, 56, 77], 65=[119, 150, 178], 14(

nivel del arbol = 2
de de{21=[51], 177=[41]}
de base{40=[13], 61=[10], 120=[45]}
de datos{1=[39, 49], 206=[101], 201=[14], 200=[44, 56], 42=[121], 230=[11], 9=[47], 108=[2
base de{16=[62], 138=[68], 206=[100, 122], 115=[63], 201=[13], 6=[78], 189=[92], 42=[61],
datos de{35=[168], 16=[64, 74], 1=[40], 206=[102], 201=[15], 212=[64], 247=[116, 127], 28=

nivel del arbol = 3
de base de{61=[11]}
de datos de{16=[64], 1=[40], 206=[102], 201=[15], 247=[116, 127]}
base de datos{16=[63], 206=[101], 201=[14], 247=[126, 203], 268=[3], 61=[12]}

nivel del arbol = 4
de base de datos{61=[12]}
base de datos de{16=[64], 206=[102], 201=[15], 247=[127]}
```

Figura 4. 7 términos latentes existentes para la colección de documentos que tienen a un elemento de $\mathbb{K}_e$

La línea con la leyenda *nivel del árbol = 1* indica que para nivel *1* se lista a los documentos que tienen ocurrencias de los términos latentes con longitud uno que aparecen en la consulta, nivel del árbol 2, indica los documentos con términos latentes de longitud dos; si se observa la consulta escrita tiene longitud tres, y un nivel máximo de cuatro, esto debido a que en la colección existen combinaciones de los términos latentes encontrados en la consulta, por ejemplo *base de datos de.*

La forma de seleccionar nodos del árbol de términos latentes, es mediante una cadena de texto que representa los nodos seleccionados con una sintaxis *nivel, nodo* repetida tantas veces sea necesario, por ejemplo si se quiere seleccionar al nodo *base de datos* se debe hacer referencia a el nivel 3 y al nodo 3. Con esto se obtiene a los nodos seleccionados de los nodos existentes: $\mathbb{K}_s \subseteq \mathbb{K}_e$.

La notación para elegir nodos no tiene la finalidad de ser didáctica o amable con el usuario, su finalidad es obtener $\mathbb{K}_s$ de $\mathbb{K}_e$.

```
Escriba las cadenas que señalen los nodos a usar
3,3
base de datos{16=[63], 206=[101], 201=[14], 247=[126, 203], 268=[3], 61=[12]}
numero de terminos latentes seleccionados = 1
**************************************************
```

Figura 4. 8 Selección de un nodo

Entonces como resultado se obtiene una lista con los documentos que tienen al término latente y las posiciones en que el término latente ocurre en el documento, como se puede ver en la **Figura 4.8 Selección de un nodo**.

Un ejemplo más, con la misma consulta pero con una selección de nodos diferente, donde $\mathbb{K}_s \subseteq \mathbb{K}_e$ se ve en la **Figura 4.9 Selección de un nivel**.

```
Escriba las cadenas que señalen los nodos a usar
1,1 1,2 1,3
de{1=[10, 13, 24, 38, 40, 48], 2=[5, 12, 17, 26, 34, 36, 69, 89,
base{1=[19], 207=[31], 206=[99, 121], 138=[67], 201=[12, 75], 5=
datos{1=[39, 49], 207=[91], 206=[101], 201=[14], 200=[44, 56, 77
numero de terminos latentes seleccionados = 3
```

Figura 4. 9 Selección de un nivel

Como se puede observar en este caso, se seleccionaron a todos los términos latentes de longitud uno, es decir, en este caso se está calculando mediante el modelo vectorial clásico, en otras palabras el modelo vectorial clásico está incluido en esta extensión, esto es de utilidad para calcular la precisión en ambos modelos para consultas iguales, trabajo que se llevo a cabo y es explicado en el capítulo siguiente.

Referencias bibliográficas y fuentes de información

[Sparck-R. & Robertson-S.]Robertson, S. E., y Sparck Jones, K. *Relevance weighting of search terms*, Journal of the American Society for Information Science, Volume 27, 1976 pp. 129–146.

[Gelbukh-A.&Sidorov-G.,2006] Gelbukh-Alexander y Sidorov-Grigori PROCESAMIENTO AUTOMÁTICO DEL ESPAÑOL CON ENFOQUE EN RECURSOS LÉXICOS GRANDES. Centro de Investigación en Computación Instituto Politécnico Nacional: México D.F. :2006 p.189 ISBN 970-36-0264-9

[Sidorov-G.,2011] Curso "Fundamentos de Recuperación de Información Textual" Centro de Investigación en Computación Instituto Politécnico Nacional: México D.F. :2011

[RIJ, 1999] Rijsbergen, C.J. Information Retrieval. Glasgow, University, 1999.

[MAN,2009] Christopher D. Manning Prabhakar Raghavan Hinrich Schütze. Introduction to Information Retrieval.
Cambridge University Press Cambridge, England :2009

[ALA, 1983] Glosario A.L.A. de Bibliotecología y Ciencias de la Información. Madrid: Díaz de Santos, 1983.

[LON, 1989] Longley, D. and Shain M. Mac Millan Dictionary of IT. London and Basingstoke: The MacMillan Press, 1989.

[MEA, 1992] Meadow, C. T. Text Information retrieval Systems. San Diego: Academic Press, 1993.

[PER, 2000] Pérez-Carballo, J. and Strzalkowski, T. 'Natural language information retrieval: progress report'. Information Processing and Management 36, 2000. p. 155-178

[GRO, 1998] Grossman, D.A. and Frieder, O. Information retrieval: algorithms and heuristics. Boston: Kluwer Academia Publishers, 1998.

[MEA, 1992] Meadow, C. T. Text Information retrieval Systems. San Diego: Academic Press, 1993.

[BLA, 1990] Blair, D.C. Language and representation in information retrieval. Amsterdam [etc.]: Elsevier Science Publishers, 1990.

[TRA, 1997] Tramullas Sáez, J. Introducción a la Documática. Zaragoza: Kronos, 1997.

[BAE, 1992] Baeza-Yates, R. and Frakes, W.B. Information retrieval : data structures & algorithms Englewood Cliffs, New Yersey : Prentice Hall, 1992 504 p. ISBN 0-13-463837-9
[BAE, 1999] Baeza-Yates, R. and Ribeiro-Neto, B. Modern information retrieval. New York : ACM Press ; Harlow [etc.] : Addison-Wesley, 1999 XX, 513 p. ISBN 0-201-39829-X

[SAL, 1983] Salton , G. and Mc Gill, M.J. Introduction to Modern Information Retrieval. New York: Mc Graw-Hill Computer Series, 1983.

[IEI, 1997] International Encyclopedia of Information & Library Science. London: Rotledge, 1997

[CRO, 1987] Croft, W. B. 'Approaches to intelligent information retrieval'. Information Proccesing & Management, 23, 4, 1987.p.24954.

[CHO, 1999] Chowdhury, G. G. Introduction to modern information retrieval. London: Library Association, 1999.

[KOR, 1997] Korfhage, R.R. Information Retrieval and Storage. New York: Wiley Computer Publisher, 1997.

[Patward S. & Riloff E.,2006] Patwardham S. & Riloff E. "Learning Domain- Specific Information Extraction Patterns from the Web". Proceedings of the ACL 2006 Workshop on Information Extraction Beyond the Document, pp. 66-73, 2006.

[Kovács L. & Ahonen-Myka H.,2001] Kovács L. & Ahonen-Myka H. "Algorithm for Maximal Frequent Sequences in Document Clustering". 3rd International Symposium of Hungarian Researchers on Computational Intelligence, pp. 89-94, 2001.

[Ahonen-Myka H.,2002] Ahonen H. "Discovery of Frequent Word Sequences in Text". Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery. London, UK, pp. 180-189, 2002.

[Porter, Martin F. 1980]. An algorithm for suffix stripping. Program 14(3):130–137.

[Salton 1989]Salton, Gerard. 1989. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley.

[Harman 1991]Harman, Donna. 1991. How effective is suffixing? JASIS 42:7–15.

[Krovetz 1995]Krovetz, Bob. 1995. Word sense disambiguation for large text databases. PhD thesis, University of Massachusetts Amherst.

[Hull 1996]Hull, David. 1996. Stemming algorithms – A case study for detailed evaluation. JASIS 47(1):70–84.

Capítulo V
Pruebas y resultados

¡Querida, realmente tengo que conseguir un lápiz más fino! No puedo en absoluto manejar éste: escribe todo tipo de cosas, sin que yo se las dicte.

Lewis Carroll, Alicia a través del espejo.

Resumen

Los ejemplos de ejecución para la implementación expuesta en el capítulo anterior, no son estadísticamente representativos, para tener una inferencia verosímil de los resultados obtenidos, se busca que el número de ejercicios sea representativo, en seguida se ejecutan estos ejercicios, se calculan las exactitudes para cada uno de estos ejercicios y se interpretan los resultados. Es necesario comentar que la Biblioteca del CIC facilito parte de su base de datos de artículos de la revista de computación y sistemas del 1997 a 2009.

Para que algo se pueda calificar debe ser medible, la extensión al modelo clásico que en este trabajo se expone, tiene como característica una exactitud mayor que el modelo clásico. Las pruebas hechas tienen como finalidad comparar la exactitud del modelo clásico contra la exactitud de la extensión. Los resultados expuestos en este capítulo están resumidos mediante gráficos y están respaldados por la ejecución de la implementación expuesta en el capítulo anterior.

5.1 Número y características de los ejercicios

La teoría de la inferencia estadística consiste en aquellos métodos con los cuales se pueden realizar inferencias o generalizaciones acerca de una población [WAL,1992], sin importar cual sea el método usado para calcular la verosimilitud de una muestra aleatoria, se sabe que el sesgo de está (la verosimilitud), tiende a cero conforme el tamaño de la muestra, se hace más grande.

La característica a medir en este conjunto de resultados, es la exactitud para evitar el sesgo en los resultados, el número de ejercicios es el total de ejercicios posibles. Para encontrar todos los ejercicios posibles, se construyo un archivo inverso con todos los términos latentes existentes en la colección, el programa encargado de generar este archivo inverso de términos latentes de cualquier longitud, es *PruebaDoceF.java* encontrado en los anexos, que implementa el seudocódigo descrito en la **Figura 3.6 Seudocódigo para generar un Archivo inverso de términos latentes de cualquier longitud**.

El archivo inverso de todos los términos latentes de cualquier longitud se encuentra en el archivo anexo *TerminosYUbicacionEnDocumentos.txt*. Después de encontrar a todos los términos latentes de cualquier longitud, se puede deducir qué consultas se pueden contestar, debido a que una consulta no encontrada en el archivo inverso de términos latentes, no es una consulta encontrada en la colección de documentos.

El número total de consultas con respuesta contenida en la colección de documentos, es 24,687 con consultas que tienen longitudes de uno a dieciocho, esta colección de consultas, es la utilizada para comparar la exactitud del modelo clásico contra la exactitud del modelo extendido.

Hacer consultas con una longitud mayor, es equivalente a la sumatoria de un conjunto de consultas menores o combinaciones de estas, debido a esto sólo se hacen ejercicios con las 24,687 consultas, contenidas en el archivo TerminosYUbicacionEnDocumentos.txt.

5.2 Características de la fuente de documentos usados para medir

La cantidad de documentos usados, es de 267 correspondientes con igual número de resúmenes de las publicaciones de la revista Computación y Sistemas en el período 1997-2009. Esta colección tiene un total de 24,687 términos diferentes, con longitudes que van de 1 a 18, distribuidos de la siguiente forma:

Número de ocurrencias y longitud de términos en las publicaciones de Computación y Sistemas.

Tabla 5.1 Número de ocurrencias y longitud de términos en las publicaciones de Computación y Sistemas

Número de términos latentes	Longitud del término latente
7,335	1
8,830	2
5,693	3
1,893	4
579	5
172	6
76	7
33	8
20	9
12	10
9	11
8	12
7	13
6	14
5	15
4	16
3	17
2	18

El contenido de los documentos usados está en el archivo de texto anexo, con nombre de archivo *Documentos.txt*. Para poder hacer un comparativo de exactitudes entre el modelo clásico y esta extensión, hacia falta un punto de referencia con respecto al cual medir, este punto de referencia es un archivo inverso de todos los términos latentes de longitudes de uno a dieciocho, generado con la colección de documentos, este archivo está guardado como texto plano en el archivo que se encuentra como anexo *TerminosYUbicacionEnDocumentos.txt*.

5.3 Cálculo de la exactitud

El modelo clásico esta incluido en el extendido, esto porque el modelo clásico vectorial toma a cada término de la consulta y lo usa para calcular la semejanza contra la colección de documentos, seleccionar a los nodos del primer nivel del árbol de términos latentes generado a partir de una consulta para calcular la semejanza contra la colección de documentos, es equivalente al modelo clásico.

La implementación del modelo extendido, igual que el modelo clásico, dan como resultado un conjunto de documentos ordenados de manera decreciente por su semejanza con la consulta usada para comparar. Nuestro punto de referencia, es el archivo inverso de términos latentes, del que se extraen todas las consultas usadas en el comparativo, el renglón correspondiente con la consulta hecha, tiene una lista con los documentos que contienen al término y las posiciones en que este ocurre.

Este punto de referencia sirve para saber qué documentos son las respuestas relevantes que el modelo regresa, ambos modelos regresan en su conjunto de resultados a estos documentos, la diferencia está en el tamaño del conjunto de documentos, en los que estos documentos relevantes están incluidos.

Probabilidad de que un documento recuperado sea relevante:

$$\text{precisión} = \frac{documentos\ relevantes\ \cap documentos\ recuperados}{documentos\ recuperados} \quad (5.1)$$

Probabilidad de que un documento relevante sea recuperado:

$$cobertura = \frac{documentos\ relevantes\ \cap documentos\ recuperados}{documentos\ relevantes} \quad (5.2)$$

Una medida clásica en estadística para calcular la exactitud es *F1score* que involucra la presicion y el cobertura conceptos que enpatan con la *probabilidad de que un documento recuperado sea relevante* y la *probabilidad de que un documento relevante sea recuperado.*

$$exactitud = 2 * \frac{\text{precisión} * cobertura}{\text{precisión} + cobertura} \quad (5.3)$$

Por ejemplo para *C* = ***bases de datos***:
El Archivo inverso de los términos latentes de longitud uno *AI* y la consulta *C* sirven para construir el siguiente árbol de términos latentes $ATL_{base\ de\ datos}$:

nivel del árbol = 0
de{1=[10, 13, 24, 38, 40, 48], … , 265=[8, 19, 38, 50, 60], 268=[2]}
bases{256=[2], … , 210=[159]}
datos{1=[39, 49], … , 61=[12]}

nivel del árbol = 1
de de{21=[51], 177=[41]}
de bases{144=[127]}

de datos{1=[39, 49], … , 268=[2]}
bases datos{140=[47]}
datos de{35=[168], … , 225=[86]}

nivel del árbol = 2
de bases de{144=[128]}
de datos de{16=[64], 1=[40], 206=[102], 201=[15], 247=[116, 127]}
bases de datos{54=[113], 22=[95], 268=[3]}

Modelo clásico
Se seleccionan los tres nodos del primer nivel del árbol de términos latentes y el resultado es una colección de 263 documentos:
263 documentos que tienen al término "de"
8 documentos que tienen al término "bases"
45 documentos que tienen al término "datos"

Si se verifica en el documento *TérminosYUbicacionesEnDocumentos.txt* en el renglón *bases de datos{54=[113], 22=[95]},* se puede saber que los únicos documentos que contienen el término latente *bases de datos* son el 54, el 22, y la consulta.

El número de documentos recuperados es 263
El número de documentos relevantes es 3
Como los documentos relevantes son un subconjunto de los recuperados
$documentos\ relevantes \cap documentos\ recuperados = 3$

$$\text{precisión} = \frac{3}{263} = 0.011406844106464$$

$$cobertura = \frac{3}{3} = 1$$

$$exactitud = 2 * \frac{0.011406844106464 * 1}{0.011406844106464 + 1} = 0.022556390977444$$

Modelo Extendido
En el modelo extendido seleccionamos al nodo correspondiente con la consulta, dando como resultado:

3 documentos que tienen al término latente "bases de datos"

Si se verifica en el documento *TérminosYUbicacionesEnDocumentos.txt* en el renglón *bases de datos{54=[113], 22=[95]},* se puede saber que los únicos documentos que contienen el término latente *bases de datos* son el 54, el 22, y la consulta.

Como los documentos relevantes son un subconjunto de los recuperados
$documentos\ relevantes \cap documentos\ recuperados = 3$

$$\text{precisión} = \frac{3}{3} = 1$$

$$cobertura = \frac{3}{3} = 1$$

$$exactitud = 2 * \frac{1 * 1}{1 + 1} = 1$$

Resultado de la comparación:

Tabla 5.2 Comparación de exactitud

Término latente	Exactitud modelo clásico	Exactitud modelo extendido
bases de datos	0.022556390977444	1

El ejercicio anterior se llevo a cabo con la consulta *bases de datos*, el documento anexo *TerminosYUbicacionEnDocumentos.txt* tiene 24,687 renglones, en cada renglón se encuentra un término latente diferente, para cada término latente diferente se ejecuto un ejercicio como el anterior, los resultados de estos ejercicios se guardaron en el archivo anexo *Exactitudes.txt*; en cada renglón, se encuentra de izquierda a derecha, la consulta usada, la exactitud del modelo clásico y la exactitud del modelo extendido.

5.4 Ejecución de la comparación

La muestra de una colección, es más representativa conforme esta muestra, aproxime su tamaño al tamaño de la colección que la contiene. Debido a esto se opto por buscar todas las consultas con respuesta en la colección de documentos. Estas consultas, junto con la lista de documentos que las contienen, están guardadas en el archivo anexo *TerminosYUbicacionEnDocumentos.txt*, éste archivo es el resultado de ejecutar el programa *PruebaDoceF.java*, también anexo.

El archivo *TerminosYUbicacionEnDocumentos.txt,* tiene dos funciones en la ejecución de la comparación de exactitud entre ambos modelos, en principio sirve para saber cuantas ejecuciones se deben hacer de la implementación, y la segunda, sirve para saber que documentos tienen a la consulta.

Para conseguir las 49,374 ejecuciones, resultantes de usar los 24,687 términos latentes del diccionario de términos del archivo inverso de términos latentes, para el modelo clásico y esos mismos 24,687 para el modelo extendido, tomando en cuenta que ejecutar la aplicación a mano, requiere de unos veinte segundos, la ejecución del ejercicio requeriría cerca de 300 horas para concluir su ejecución y se correría el riesgo de tener errores al copiar el término latente del diccionario de términos.

Debido a estos inconvenientes se opto por escribir una aplicación, que tomará a cada término latente del diccionario de términos del archivo de términos latentes e instanciará un par de implementaciones, una para el modelo clásico y otra para el modelo extendido.
La implementación referida, es la explicada en el capítulo implementación y el proyecto en el que ésta implementación es empotrada, para poder ejecutar de manera automática el conjunto de consultas es *Enero,* proyecto anexo a este documento, Ésta implementación se puede ver como dos partes. La primera, donde se calcula la semejanza de la consulta contra la colección de documentos, y la segunda donde se calcula la exactitud de cada modelo.

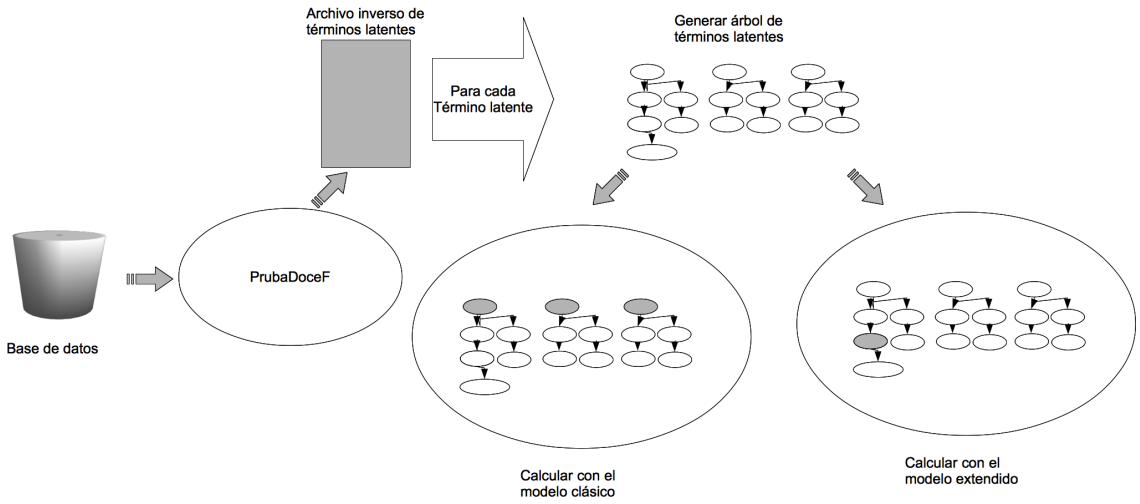


Figura 5.1 Cálculo de la semejanza con ambos modelos para cada Término Latente

La implementación explicada en el capitulo anterior tiene un conjunto de adiciones para llevar a cabo el conjunto de ejercicios, estas adiciones están resumidas de manera gráfica en la **Figura 5.1,** una de las adiciones hechas fue la inclusión de una instancia de la clase *PruebaDoceF.java,* que como ya se menciono, sirve para generar un archivo inverso de términos latentes de cualquier longitud. Aunque en los anexos se incluye un archivo inverso de todos los términos latentes de cualquier longitud, dentro de la implementación se usa una instancia de la clase *PruebaDoceF* que lo genero y no la lectura de el archivo inverso, debido a que es mas rápido generar el archivo inverso que leerlo, en caso de que la colección de documentos cambie será automáticamente corregido el archivo inverso y por ultimo, porque el tamaño de este archivo es tan pequeño (893KB) que fácilmente se puede contener en memoria.

Este archivo inverso de términos latentes, es visto internamente como una tabla hash que anida a otra y esta a su vez anida a una lista, se opto por esta implementación, por su rapidez y facilidad de programación. Esta estructura es usada para consultar de ella a los términos que involucre la selección de nodos del árbol de términos latentes.

También se modifico la clase *SeleccionarNodos,* para que el sistema seleccionara en el caso del modelo clásico, a todos los nodos del primer nivel y para el caso de el modelo extendido, al nodo correspondiente con la consulta, esto permite usar para cada término latente diferente, un par de instancias de la implementación que recibirán un conjunto de nodos diferentes seleccionados, pero que representan la misma consulta tanto para el modelo clásico como para el extendido.

La ultima inclusión fue la clase *Eficiencia*, encargada de calcular la exactitud para cada término en ambos modelos y de escribir los resultados en un archivo, el conjunto de resultados para este ejercicio, esta guardado en el documento *Exactitudes.txt*, donde cada renglón compara la exactitud del modelo clásico, contra la exactitud del modelo extendido, tomando como consulta a cada uno de los 24,687 términos listados en el archivo.

La clase *Eficiencia,* calcula la exactitud de un modelo, primero obteniendo como argumento un conjunto de abstracciones de los documentos, que el modelo da como resultado, cada elemento de este conjunto de abstracciones, sirve para determinar que grado de semejanza el modelo le asigno al documento que esa abstracción representa, es decir la clase obtiene como argumentos iniciales dos conjuntos de abstracciones de documentos.

La segunda utilidad del archivo inverso de términos latentes para ambos modelos, es saber que documentos tienen a una consulta. Debido a que las consultas hechas en este ejercicio son elementos del diccionario de términos del archivo inverso, de términos latentes, el archivo inverso de términos latentes, sirve para saber que documentos tienen a la consulta.

La clase *Eficiencia,* obtiene a la consulta que es un atributo con ámbito de clase de la implementación, esta consulta es buscada en la representación interna del archivo inverso de términos latentes y se extrae el conjunto de números de identificación de los documentos, en que la consulta aparece, este conjunto de documentos es útil debido a que la exactitud de un modelo, se calcula usando el número de documentos relevantes.

El cálculo de la exactitud de un modelo respecto a una consulta, se hace dentro de una instancia de clase de tipo *Eficiencia*, el conjunto de abstracciones de documentos que obtiene como argumento inicial, sirve para calcular la exactitud de este modelo respecto a una consulta en la colección de documentos. Como es fácil deducir el proceso para calcular la exactitud, es usado dos veces, una para el modelo clásico y otra para el modelo extendido.

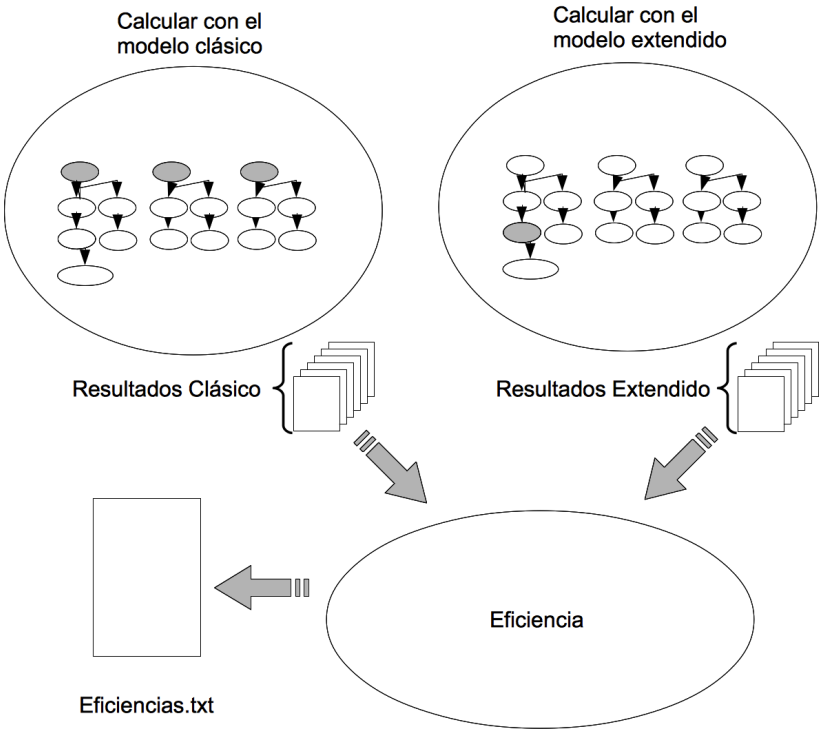


Figura 5. 2 Cálculo de la exactitud con ambos modelos para cada Término Latente

La figura anterior (**Figura 5.2**), resume la tarea de calcular la exactitud de ambos modelos tomando como argumentos los resultados de cada modelo, es decir un par de abstracciones de documentos y el archivo inverso de términos latentes del que se extrae el conjunto de índices de documentos que contienen a la consulta. Una vez calculada la exactitud de ambos modelos, se imprime una yuxtapuesta a otra, precedidas por la consulta hecha, esto para cada renglón en un documento de texto plano de nombre *Exactitudes.txt,* que se puede encontrar en los anexos de este trabajo.

5.5 Comparación del modelo clásico contra la extensión

Tanto el modelo clásico como el modelo extendido en sus resultados tienen a los documentos que se están buscando, cada documento perteneciente al conjunto de documentos que un modelo da como respuesta a una consulta, cada uno de esos documentos, tiene asignado un valor de semejanza, este valor de semejanza es el discriminador, mediante el que se ordenan primero los documentos con un valor de semejanza mayor.

El punto de comparación entre el modelo clásico y el modelo extendido en este trabajo, es la exactitud de los resultados que dan los modelos, ya se explicó que la confiabilidad de los resultados en una inferencia estadística es mayor conforme crece el número de elementos contenidos en la muestra usada para el estudio, por tal razón se aplican todos los ejercicios posibles para la colección de documentos, con que se trata en éste experimento. Ya se explicó también, que se necesita de un archivo inverso de términos latentes de cualquier longitud, para la colección de documentos a usarse como punto de referencia y poder medir la exactitud de los modelos.

Éste apartado tiene como finalidad describir el conjunto de experimentos hechos, que servirán para respaldar la interpretación que se dé en ellos, el número de experimentos se cuenta en miles, lo que hace difícil mostrarlos en este documento, el resultado de ejecutar todos los experimentos del ejercicio, es un archivo de texto plano, en el que se imprime la consulta que se aplica a ambos modelos, el árbol de términos latentes que esta consulta genera y los resultados arrojados por ambos modelos, específicamente este conjunto de resultados, es un valor de semejanza y un documento al que se le asignan datos como términos usados y el número de documentos con el término, todo esto se da para cada consulta, que se determina por el número de términos latentes encontrados en el archivo inverso de términos latentes.

El resultado de este conjunto de ejecuciones, se guarda en un documento de texto plano de nombre *Resultado.txt*, el tamaño de este documento es de 4.8GB, este archivo por su tamaño no se anexa al presente documento, pero si se anexa la base de datos *CyS* y el programa *Enero* que lo genera.

El tamaño de éste documento se debe a que para cada consulta hecha, se imprimen los valores que servirán para la comprobación de su correcto cálculo así para saber si el peso ponderado de un término es correcto, se cuenta con valores como: número de ocurrencias de ese término en ese documento, número de documentos que tienen al término, y esto se hace para cada término involucrado en el cálculo de la semejanza, además de imprimir el documento en cuestión, con la finalidad de verificar el número de ocurrencias que se afirman.

El documento *Resultado.txt,* sirve para verificar el buen funcionamiento de la implementación que ejecuta el conjunto de experimentos que componen al ejercicio que se describe. Los resultados guardados en el documento Resultados, corresponden con valores de semejanza asignados a los documentos involucrados, no a valores de exactitud, que es la característica importante en este capítulo, debido a que la exactitud es el punto a comparar de ambos modelos.

Si se tienen los valores de exactitud que ambos modelos arrojan para una consulta y se tiene un archivo inverso de términos latentes de cualquier longitud, obtener la exactitud es posible, en este ejercicio, la ejecución de este cálculo se da en el programa *Enero*, que guarda los resultados en el documento anexo *Exactitudes.txt*.

5.6 Resultados de la comparación del modelo clásico contra la extensión

5.6.1 Consultas de longitud 1

La consulta *programas* aparece en los documentos 263, 66, 148, 106 y en la consulta:

Tanto en el modelo clásico como en el extendido, se obtiene como resultado a los cinco documentos, esto debido a que seleccionar a todos los nodos del primer nivel del árbol de términos latentes, es seleccionar al nodo único y seleccionar al nodo correspondiente con la consulta, significa también seleccionar al nodo único.
Por *Formula 5.1-5.3* se calcula la exactitud para ambos modelos:

$$precisión = \frac{5}{5} = 1$$

$$cobertura = \frac{5}{5} = 1$$

$$exactitud = 2 * \frac{1 * 1}{1 + 1} = 1$$

Tabla 5.3 Cálculo de la exactitud con ambos modelos para el término latente *programas*

Término latente	Exactitud modelo clásico	Exactitud modelo extendido
programas	1	1

El anterior es un ejemplo para una de las 7335 consultas con longitud uno, que se usaron para comparar al modelo clásico con el extendido, lo que dió 7335 resultados para el modelo clásico y 7335 para el modelo extendido, Para hacer fácil la representación de estos resultados se promediaron, dando como resultado la siguiente tabla:

Tabla 5.4 Cálculo de la exactitud con ambos modelos para el conjunto de términos latentes de longitud 1

Longitud de las consultas	Exactitud modelo clásico	Exactitud modelo extendido
1	1	1

Esto significa que cuando la consulta tiene una palabra de longitud uno, tanto el modelo clásico como el modelo extendido tienen un 100% de exactitud.

5.6.2 Consultas de longitud 2

La consulta *muchos sistemas* aparece en los documentos 76 y en la consulta:

En el modelo clásico se obtiene como resultado a 70 documentos con el término latente *sistemas* y 11 con el término latente *muchos*, esto da como resultado un conjunto de 75 documentos, en este conjunto se encuentran los dos documentos relevantes.
Por *Formula 5.1-5.3* se calcula la exactitud para el modelo extendido.

$$precisión = \frac{2}{75} = 0.026666666666667$$

$$cobertura = \frac{2}{2} = 1$$

$$exactitud = 2 * \frac{0.026666666666667 * 1}{0.026666666666667 + 1} = 0.051948051948052$$

En el modelo extendido se obtiene como resultado a los dos documentos 76 y la consulta.

Por *Formula 6.1-6.3* se calcula la exactitud para el modelo extendido:

$$precisión = \frac{2}{2} = 1$$

$$cobertura = \frac{2}{2} = 1$$

$$exactitud = 2 * \frac{1 * 1}{1 + 1} = 1$$

Tabla 5.5 Cálculo de la exactitud con ambos modelos para el término latente *muchos sistemas*

Término latente	Exactitud modelo clásico	Exactitud modelo extendido
muchos sistemas	0.051948051948052	1

El anterior es un ejemplo para una de las 8,830 consultas con longitud dos, que se usaron para comparar al modelo clásico con el extendido, lo que dio 8,830 resultados para el modelo clásico y 8,830 para el modelo extendido, para hacer fácil la representación de estos resultados, se promediaron, dando como resultado la siguiente tabla:

Tabla 5.6 Cálculo de la exactitud con ambos modelos para los términos latentes de longitud 2

Longitud de las consultas	Exactitud modelo clásico	Exactitud modelo extendido
2	0.109257637	1

Esto significa que cuando la consulta tiene dos palabras de longitud, el modelo clásico tiene un 10.9% de exactitud y el modelo extendido tiene un 100% de exactitud.

5.6.3 Consultas de longitud 3

La consulta *compuestos por subsistemas* aparece en los documentos 13 y en la consulta:

En el modelo clásico se obtiene como resultado a 2 documentos con el término latente *subsistemas*, 4 con el término latente *compuestos* y 142 con el término latente *por*, esto da como resultado un conjunto de 142 documentos, en este conjunto se encuentran los dos documentos relevantes.
Por *Formula 5.1-5.3* se calcula la exactitud para el modelo extendido:

$$precisión = \frac{2}{142} = 0.014184397163121$$

$$cobertura = \frac{2}{2} = 1$$

$$exactitud = 2 * \frac{0.014184397163121 * 1}{0.014184397163121 + 1} = 0.027972027972028$$

En el modelo extendido se obtiene como resultado a los dos documentos 13 y la consulta.

Por *Formula 5.1-5.3* se calcula la exactitud para el modelo extendido.

$$precisión = \frac{2}{2} = 1$$

$$cobertura = \frac{2}{2} = 1$$

$$exactitud = 2 * \frac{1 * 1}{1 + 1} = 1$$

Tabla 5.7 Cálculo de la exactitud con ambos modelos para el término latente *compuestos por subsistemas*

Término latente	Exactitud modelo clásico	Exactitud modelo extendido
compuestos por subsistemas	0.027972027972028	1

El anterior es un ejemplo para una de las 5,693 consultas con longitud tres, que se usaron para comparar al modelo clásico con el extendido, lo que dio 5,693 resultados para el modelo clásico y 5,693 para el modelo extendido, Para hacer fácil la representación de estos resultados se promediaron, dando como resultado la siguiente tabla:

Tabla 5.8 Cálculo de la exactitud con ambos modelos para los términos latentes de longitud 3

Longitud de las consultas	Exactitud modelo clásico	Exactitud modelo extendido
3	0.02286265	1

Esto significa que cuando la consulta tiene tres palabras de longitud, el modelo clásico tiene un 2.28% de exactitud y el modelo extendido tiene un 100% de exactitud.

5.7 Resultados de la comparación del modelo para consultas de longitudes de uno a dieciocho

Como en el apartado anterior, se pueden observar las consultas con longitudes diferentes a uno que dan mejores resultados en cuanto a exactitud para el modelo extendido comparado contra el modelo clásico, para resumir los resultados en este apartado, se muestra de manera conjunta el promedio de exactitudes para los resultados obtenidos con ambos modelos, agrupados por la longitud de la consulta (**Tabla 5.9**).

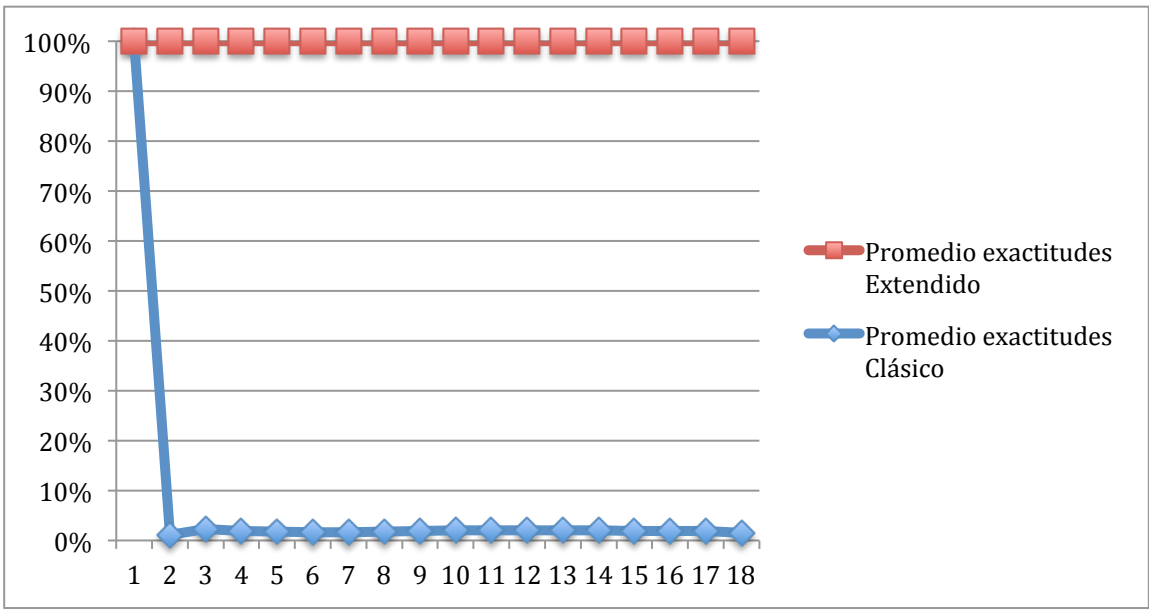
Tabla 5.9(1/2) Cálculo de la exactitud con ambos modelos para términos latentes de longitudes de uno a dieciocho

Longitud de las consultas	Promedio exactitudes clásico	Promedio exactitudes extendido
1	1	1
2	0.109257637	1
3	0.02286265	1
4	0.019230792	1
5	0.017713049	1
6	0.017105318	1
7	0.017261422	1
8	0.018666947	1
9	0.019290451	1
10	0.020493514	1
11	0.200977066	1

Longitud de las consultas	Promedio exactitudes clásico	Promedio exactitudes extendido
12	0.020769011	1
13	0.020501511	1
14	0.020144845	1
15	0.019645512	1
16	0.018896512	1
17	0.018896512	1
18	0.015151515	1

Tabla 5.9(2/2) Cálculo de la exactitud con ambos modelos para términos latentes de longitudes de uno a dieciocho

Los resultados de la **Tabla 5.8,** pueden ser vistos como porcentajes de exactitud en cada modelo, en la **Grafica 5.3** se aprecia esto. El rango de resultados para el modelo clásico es muy reducido, por lo que se opto por ampliar estos valores en la **Gráfica 5.4**



Grafica 5.3 Exactitud con ambos modelos para los términos latentes de longitudes de uno a dieciocho.

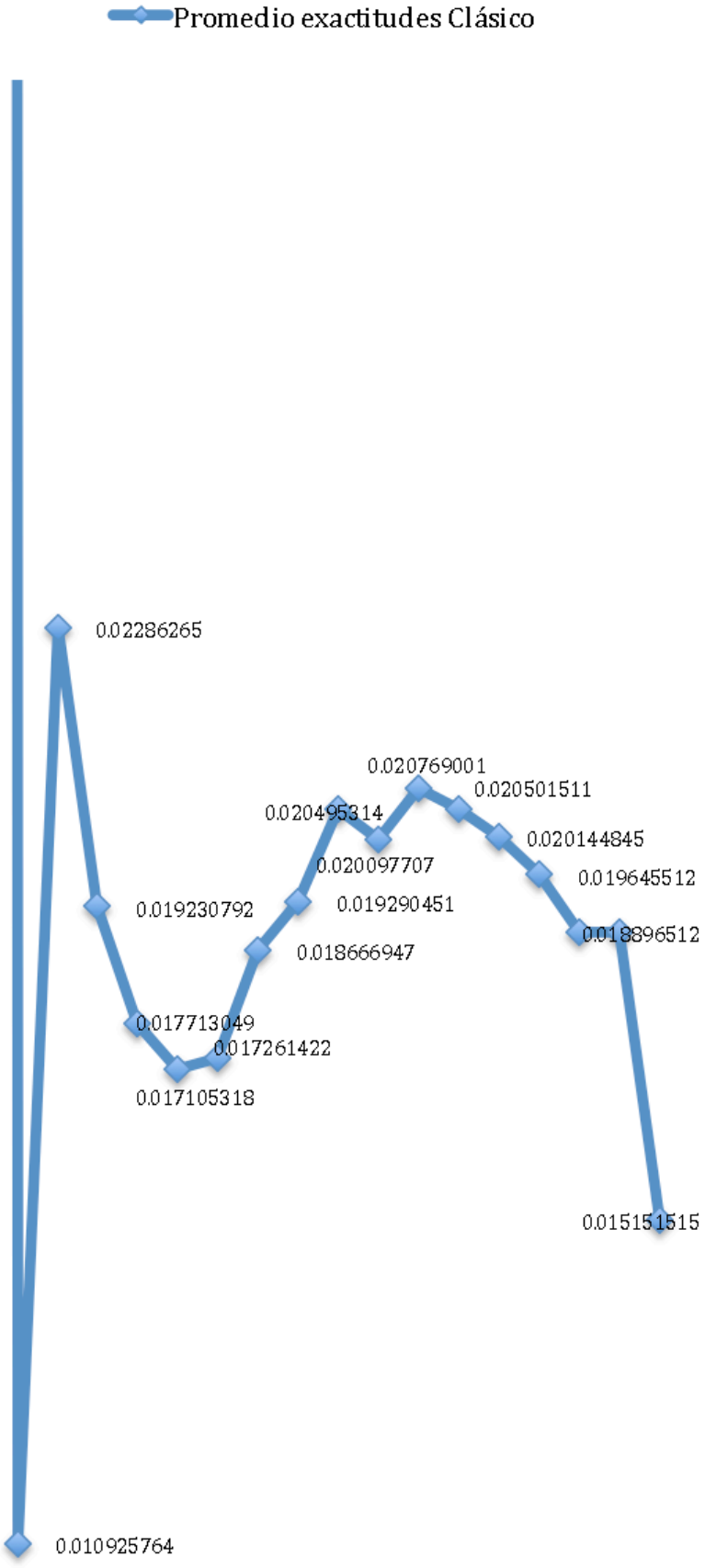
Graficar el archivo completo daría un gráfico con 24687 columnas, una representación difícil de leer; La gráfica anterior representa los valores promedio para cada conjunto de términos clasificados por su longitud.

Lo que se guarda en el archivo *Exactitudes.txt* que se uso para generar la gráfica, es un conjunto de líneas donde cada una dice: Cuál es la longitud de la consulta usada, la consulta y un par de valores numéricos, el primero representa la exactitud del modelo clásico y el segundo la exactitud del modelo extendido.

Este archivo contiene los resultados de calcular la exactitud del modelo clásico y del extendido, es el resultado de la ejecución del ejercicio que contiene a los experimentos diferentes y posibles para está colección de documentos, el nombre asignado en los anexos a este archivo es *Exactitudes.txt*, y puede ser obtenido mediante la ejecución del programa *Enero*.

En la **Gráfica 5.3,** se muestra la diferencia en los resultados obtenidos con el modelo clásico y con la extensión, lo graficado es el porcentaje de exactitud obtenido como

promedio para los términos, agrupados por longitudes, la exactitud del modelo clásico para consultas superiores a uno, no rebasa el diez por ciento de exactitud, como la gráfica 5.3 tiene un rango de uno a cien, se opto por graficar por separado los valores obtenidos con el modelo clásico, para observarse mejor, lo que da como resultado la **Gráfica 5.4**.



Gráfica 5.4 Exactitud del modelo clásico para los términos latentes de longitudes de dos a dieciocho

Anexo 1 Código para construir un árbol de términos latentes

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

public class ConstruirATL {

   ArrayList<Nivel> arbol = new ArrayList<Nivel>();
   HashMap<String, HashMap<Integer, ArrayList<Integer>>> postingPalabrasUtiles =
         new HashMap<String, HashMap<Integer, ArrayList<Integer>>>();

   ConstruirATL(HashMap<String, HashMap<Integer, ArrayList<Integer>>>
postingPalabras,
         ArrayList<String> Query) {
      obtenerPalabrasUtiles(postingPalabras, Query);
      primerNivel(postingPalabrasUtiles);
}

   public void primerNivel(HashMap<String, HashMap<Integer, ArrayList<Integer>>>
terminos) {
      Nivel primero = new Nivel();
      Iterator itTerm = terminos.entrySet().iterator();
      while (itTerm.hasNext()) {
         Map.Entry documentoYApariciones = (Map.Entry) itTerm.next();
         Nodo tmp = new Nodo();
         tmp.terminoLatente = (String) documentoYApariciones.getKey();
         tmp.documentosYPosiciones =
               (HashMap<Integer, ArrayList<Integer>>)
documentoYApariciones.getValue();
         primero.hermanos.add(tmp);
      }
      arbol.add(primero);
      Nivel nivelCero = arbol.get(0);
      insertarNivel(primero, nivelCero);
   }
public void insertarNivel(Nivel nivel, Nivel nivelCero) {
```

```
        Nivel nuevoNivel = new Nivel();

        for (Nodo n : nivel.hermanos) {
            for (Nodo nCero : nivelCero.hermanos) {
                HashMap<Integer, ArrayList<Integer>> parDocumentosListas =
                        new HashMap<Integer, ArrayList<Integer>>();

                Iterator itIdDoc = n.documentosYPosiciones.entrySet().iterator();
                while (itIdDoc.hasNext()) {
                    Map.Entry elemento = (Map.Entry) itIdDoc.next();
                    ArrayList<Integer> lista = new ArrayList<Integer>();
                    if (nCero.documentosYPosiciones.containsKey(elemento.getKey())) {
                        Integer documento = (Integer) elemento.getKey();
                        for (Integer indice : (ArrayList<Integer>) elemento.getValue()) {
                            Integer siguiente = indice + 1;
                            if (((ArrayList<Integer>)

(nCero.documentosYPosiciones.get((Integer)elemento.getKey())))
                                    .contains(siguiente)) {
                                lista.add(siguiente);
                            }
                        }
                    }
                    if (!lista.isEmpty()) {
                        parDocumentosListas.put((Integer) elemento.getKey(), lista);
                    }
                }
                if (!parDocumentosListas.isEmpty()) {
                    Nodo tmp = new Nodo();
                    tmp.documentosYPosiciones = parDocumentosListas;
                    tmp.terminoLatente = n.terminoLatente + " " + nCero.terminoLatente;
                    nuevoNivel.hermanos.add(tmp);
                }
            }
        }
        if (nuevoNivel.hermanos.size()>0) {
            arbol.add(nuevoNivel);
            insertarNivel(nuevoNivel, nivelCero);
        }
    }

    public void obtenerPalabrasUtiles(
            HashMap<String, HashMap<Integer, ArrayList<Integer>>> postingPalabras,
            ArrayList<String> Query) {
        for (String palabra : Query) {
            if (postingPalabras.containsKey(palabra)) {
                postingPalabrasUtiles.put(palabra, postingPalabras.get(palabra));
            }
        }
    }
```

```
}

class Nodo {

   String terminoLatente;
   HashMap<Integer, ArrayList<Integer>> documentosYPosiciones;

   public String toString() {
      return terminoLatente + documentosYPosiciones;
   }
}

class Nivel {

   ArrayList<Nodo> hermanos = new ArrayList<Nodo>();

   public String toString() {
      String cad = new String();
      for (Nodo nod : hermanos) {
         System.out.println(nod);
      }

      return cad;
   }
}
```