Introducción a la Recuperación de Información

Conceptos, modelos y algoritmos básicos

Nota al revisor

Esta primera versión es un borrador que no cuenta con el proceso final de edición. Esperamos tener la asistencia de un profesional del diseño gráfico a los efectos de normalizar el estilo visual y producir los gráficos en un formato estandarizado.

Por otro lado, también comunicamos que – cumplidas las tareas anteriores – se realizará la corrección gramatical y ortográfica por un profesional del área.

Además, se está construyendo un sitio web con material adicional para docentes y alumnos al presente libro. Se incluirán porciones de código, colecciones de prueba, direcciones útiles, entre otros.

Introducción a la Recuperación de Información

Conceptos, modelos y algoritmos básicos

Gabriel H. Tolosa y Fernando R.A. Bordignon
Universidad Nacional de Luján, Argentina

Laboratorio de Redes de Datos
División Estadística y Sistemas
Departamento de Ciencias Básicas
Universidad Nacional de Luján

Información de contacto con los autores

Lic. Gabriel H. Tolosa – tolosoft@unlu.edu.ar
Lic. Fernando R.A. Bordignon – bordi@unlu.edu.ar

Universidad Nacional de Luján
Departamento de Ciencias Básicas
División Estadística y Sistemas
Laboratorio de Redes de Datos

Cruce de Rutas 5 y 7 – 6700 – Luján
Buenos Aires – Argentina

Índice de Contenido

Prólogo

A partir de la expansión y consolidación de Internet, como medio principal de comunicación electrónica de datos, se ha puesto a disposición de casi toda la humanidad una importante cantidad de información de todo tipo. A los efectos de aprovechar todo este potencial de información, es necesario poseer accesos que permitan que la tarea de recuperación sea eficiente y efectiva; de esto último es lo que trata este libro.

Nuestro objetivo principal es presentar una bibliografía inicial que acompañe a un curso universitario de "Introducción a la Recuperación de Información". Si bien hay excelente bibliografía sobre el área[1], creemos que es un aporte pedagógico introducir los conceptos básicos a partir de un texto escrito en español, con abundancia de ejemplos y ejercitación original. Luego, el alumno profundizará y formalizará los temas utilizando textos de la bibliografía clásica.

Por otro lado, este libro puede ser una buena herramienta para el profesional informático dado que le acerca – de forma directa – los conceptos fundamentales y modelos básicos del área.

En la actualidad, el tema de la recuperación de información se encuentra en pleno crecimiento y – si bien – su enseñanza se aborda en seminarios y talleres extracurriculares, consideramos que en los próximos años se evaluará su incorporación definitiva como tema de grado en diferentes carreras. Esto último impulsa – aún más – la necesidad de contar con material de trabajo para docentes y alumnos.

En cuanto a las fuentes, se ha incluido tanto la bibliografía clásica como trabajos de investigación actualizados que presentan la evolución de los temas cubiertos. Además, se proponen lecturas complementarias que ayuden a profundizar y a ver campos de aplicación de la recuperación de información.

Agradecimientos

A Pablo J. Lavallén por su participación en la autoría del Anexo 6 correspondiente a la descripción de la herramienta "Full Text Search" en el motor de base de datos MySQL.

Al Profesor Pedro Hernandez por su continuo apoyo y ejemplo.

Gabriel H. Tolosa y Fernando R.A. Bordignon
Universidad Nacional de Luján
Abril de 20075

[1] De autores tales como Baeza Yates, Frakes y Grossman, entre otros.

Capítulo 1

El entorno de la recuperación de información

Históricamente, el hombre ha necesitado de medios sobre los cuales representar todo acerca del mundo que lo rodea y de reflejar – de alguna manera – su evolución. La escritura ha sido el mecanismo "tradicional" y fundamental que soporta su conocimiento en el tiempo.

Esta misma evolución ha facilitado la existencia de diferentes medios de representación de la escritura, llegando hasta nuestros días donde la información se representa digitalmente y es posible su almacenamiento y su distribución masiva en forma simple y rápida, a través de redes de computadoras. La digitalización abrió nuevos horizontes en las formas en que el hombre puede tratar con la información que produce.

De igual manera, el volumen de información existente crece permanentemente y adquiere diferentes formas de representación, desde simples archivos de texto en una computadora personal o un periódico electrónico hasta librerías digitales y espacios mucho más grandes y complejos como la web. Algunos investigadores han planteado que – desde hace varios años – existe un fenómeno denominado "sobrecarga de información" [35] debido a que el volumen y la disponibilidad hacen que los usuarios no cuenten con suficiente tiempo físico para "procesar" todo el cúmulo de medios a su alcance [9].

Entonces, resulta importante tratar con toda esa información disponible electrónicamente para que pueda servir a diferentes personas (usuarios) en diferentes situaciones. Esto plantea un desafío interesante: hay importantes volúmenes de información y hay usuarios que se pueden beneficiar de alguna manera con la posibilidad de acceder a ésta, por lo tanto, cómo poder unir preguntas con respuestas, necesidades de información con documentos, consultas con resultados? Bien, en las ciencias de la computación existe un área, la Recuperación de Información (*Information Retrieval*), que estudia y propone soluciones al escenario presentado, planteando modelos, algoritmos y heurísticas.

La Recuperación de Información (RI) no es un área nueva, sino que se viene desarrollando desde finales de la década de 1950. Sin embargo, en la actualidad adquiere un rol más importante debido al valor que tiene la información. Se puede plantear que disponer o no de la información justa en tiempo y forma puede resultar en el éxito o fracaso de una operación. De aquí, la importancia de los Sistemas de Recuperación de Información (SRI) que pueden manejar – con ciertas limitaciones – estas situaciones de manera eficaz y eficiente.

Pero, ¿Qué se entiende concretamente por "Recuperación de Información"? Para Ricardo Baeza-Yates y otros [2] "*la Recuperación de Información trata con la representación, el almacenamiento, la organización y el acceso a ítems de información*".

Años antes, Salton [50] propuso una definición amplia que plantea que el área de RI "*es un campo relacionado con la estructura, análisis, organización, almacenamiento, búsqueda y recuperación de información*".

Cabe aclarar que en las definiciones anteriores los elementos de información son no estructurados, tales como documentos de texto libre (por ejemplo, un archivo de texto que contenga La Biblia) ó semi-estructurados, como lo son las páginas web.

Croft [16] 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.*". Por otro lado, Korfhage [25] definió la RI como "*la localización y presentación a un usuario de información relevante a una necesidad de información expresada como una pregunta*"

Ciertamente, es un área amplia, donde se abarcan diferentes tópicos, algunos computacionales como el almacenamiento y la organización; y otros relacionados con el lenguaje y los usuarios como la representación y la recuperación propiamente dicha.

Nótese que Croft y Korfhage plantean explícitamente el rol del usuario como fuente de consultas y destinatario de las respuestas. Por lo tanto, de manera más genérica, podemos plantear que la recuperación de información intenta resolver el problema de "**encontrar y rankear documentos relevantes que satisfagan la necesidad de información de un usuario, expresada en un determinado lenguaje de consulta**". Sin embargo, existe un problema que dificulta sobremanera esta tarea y consiste en poder "compatibilizar" y comparar el lenguaje en que está expresada tal necesidad de información y el lenguaje de los documentos.

1.1 – La problemática de la RI

De forma general – según Baeza-Yates [2] – el problema de la RI puede ser estudiado desde dos puntos de vista: el computacional y el humano. El primer caso tiene que ver con la construcción de estructuras de datos y algoritmos eficientes que mejoren la calidad de las respuestas. El segundo caso corresponde al estudio del comportamiento y de las necesidades de los usuarios.

Si se analiza la problemática de la RI desde un alto nivel de abstracción (Figura 1) podemos establecer que:

- Existe una colección de documentos que contienen información de interés (sobre uno o varios temas)

- Existen usuarios con necesidades de información, quienes las plantean al SRI en forma de una consulta (en inglés, *query*. En adelante, ambas palabras se utilizarán indistintamente)

- Como respuesta, el sistema retorna – de forma ideal – referencias a documentos "relevantes", es decir aquellos que satisfacen la necesidad expresada, generalmente en forma de una lista rankeada.

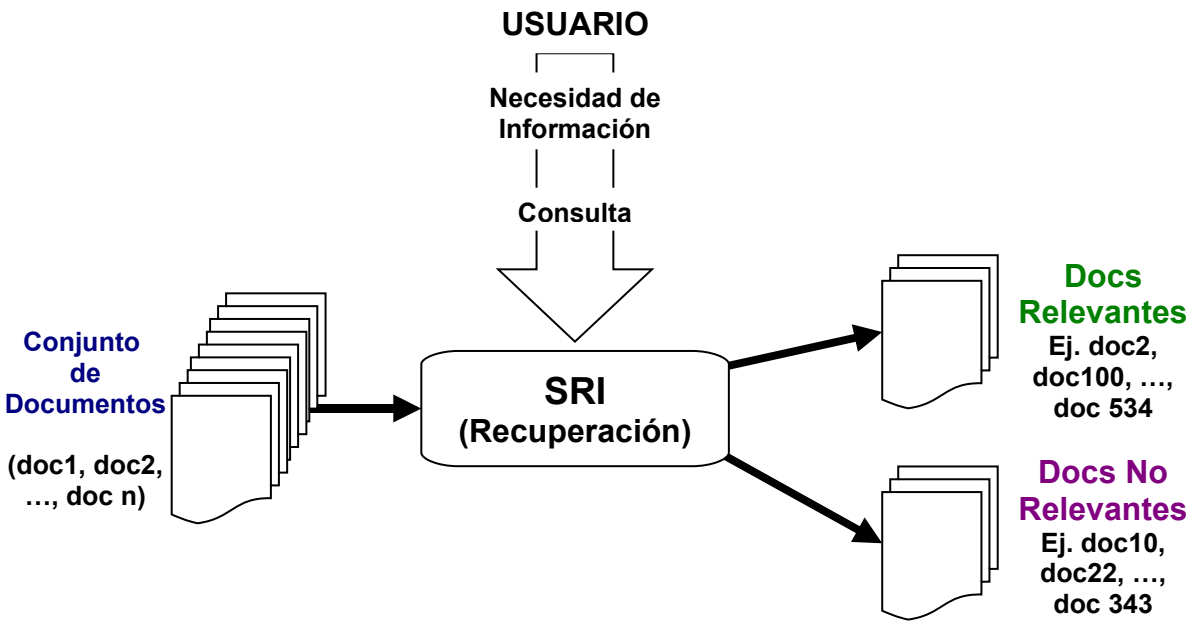


Figura 1 – La problemática de la RI

Planteamos que la respuesta "ideal" de un SRI está formada solamente por documentos relevantes a la consulta, pero – en la práctica – esta no es aún alcanzable. Esto se debe a que – entre otros motivos – existe el problema de compatibilizar la expresión de la necesidad de información y el lenguaje y de los documentos. Además, hay una carga de subjetividad subyacente y depende de los usuarios. Entonces, el SRI recupera la mayor cantidad posible de documentos relevantes, minimizando la cantidad de documentos no relevantes (ruido) en la respuesta. En términos de eficiencia, se plantea la idea de ***precisión*** de la respuesta, es decir, cuando más documentos relevantes contenga el conjunto solución (para una consulta dada), más preciso será.

Para cumplir con sus objetivos, un SRI debe realizar algunas tareas básicas, las cuales se encuentran – fundamentalmente – planteadas en cuestiones computacionales, a saber:

- Representación lógica de los documentos y – opcionalmente – almacenamiento del original. Algunos sistemas solo almacenan porciones de los documentos y otros lo hacen de manera completa.

- Representación de la necesidad de información del usuario en forma de consulta.

- Evaluación de los documentos respecto de una consulta para establecer la relevancia de cada uno.

- Ranking de los documentos considerados relevantes para formar el "conjunto solución" o respuesta.

- Presentación de la respuesta al usuario.

- Retroalimentación o refinamiento de las consultas (para aumentar la calidad de la respuesta)

En la figura 2 se puede apreciar con mayor detalle la arquitectura básica de un SRI, el tratamiento de los documentos y la interacción con el usuario. Aquí se ven algunos componentes que no se habían mencionado hasta el momento.

Como podemos observar, se parte de un conjunto de documentos de texto, los cuales están compuestos por sucesiones de palabras que forman estructuras gramaticales (por ejemplo, oraciones y párrafos). Tales documentos están escritos en lenguaje natural y expresan ideas de su autor sobre un determinado tema. El conjunto de todos los documentos con los que se trata y sobre los que se deben realizar operaciones de RI se denomina **corpus**, **colección** o **base de datos textual o documental**. Para poder realizar operaciones sobre un corpus, es necesario obtener primero una **representación lógica** de todos sus documentos, la cual puede consistir en un conjunto de términos, frases u otras unidades (sintácticas o semánticas) que permitan – de alguna manera – caracterizarlos. Por ejemplo, la representación de los documentos mediante un conjunto de sus términos se la conoce como "bolsa de palabras" (*bag of words*).

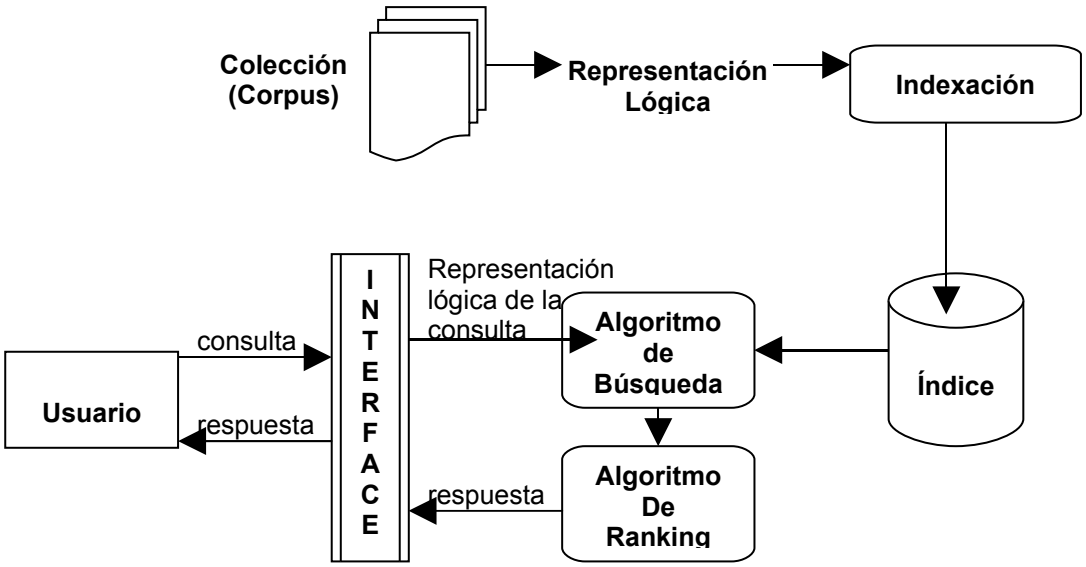


Figura 2 – Arquitectura básica de un SRI

A partir de la representación lógica de los documentos, existe un proceso (**indexación**) que llevará a cabo la construcción de estructuras de datos (normalmente denominadas **índices**) que la almacene. Estas estructuras darán luego soporte para búsquedas eficientes. Es importante destacar que una vez construidos los índices, los documentos del corpus pueden ser eliminados del sistema ya que éste retornará las referencias a los mismos debido a que cuenta con la información necesaria para hacerlo. En tal caso, el usuario será el encargado de localizar el documento para consultarlo. A los sistemas que funcionan bajo este modelo se los denomina "sistemas referenciales", en contraste con los que sí almacenan y mantienen los documentos denominados "sistemas documentales" [39]. Un ejemplo de sistemas referenciales son algunos de los motores de búsqueda web, que retornan una lista de *urls* a los documentos, como – por ejemplo –

Altavista[2]. Un caso particular es el motor de búsqueda Google[3] el cual – en algunos casos – almacena en memoria caché el documento completo, el cual puede ser consultado durante cierto tiempo, incluso si ha desaparecido del sitio original.

El **algoritmo de búsqueda** acepta como entrada una expresión de consulta o *query* de un usuario y verificará en el índice cuáles documentos pueden satisfacerlo. Luego, un algoritmo de ranking determinará la relevancia de cada documento y retornará una lista con la respuesta. Se establece que el primer ítem de dicha lista corresponde al documento más relevante respecto de a la consulta y así sucesivamente en orden decreciente.

La **interfase** de usuario permite que éste especifique la consulta mediante una expresión escrita en un lenguaje preestablecido y – además – sirve para mostrar las respuestas retornadas por el sistema.

Si bien hasta aquí se planteó la tarea básica de la RI y la arquitectura general de un SRI, el área es muy amplia y abarca diferentes tópicos. En general, un SRI no entrega una respuesta directa a una consulta, sino que permite localizar referencias a documentos que pueden contener información útil. Pero éste es solo uno de los aspectos del área de RI en la actualidad, ya que se ha atacado el problema con una perspectiva más amplia, proponiendo y desarrollando estrategias y modelos para mejorar y aumentar la funcionalidad de los SRI. Entre otras, la RI abarca tópicos como:

- Modelos de Recuperación: La tarea de la recuperación puede ser modelada desde distintos enfoques, por ejemplo la estadística, el álgebra de boole, el álgebra de vectores, la lógica difusa, el procesamiento del lenguaje natural y demás.

- Filtrado y Ruteo: Es un área que permite la definición de perfiles de necesidades de información por parte de usuarios y ante el ingreso de nuevos documentos al SRI, se los analiza y se lo reenvía para quienes se estima que van a ser relevantes.

- Clasificación: Aquí se realiza la rotulación automática de documentos de un corpus en base a clases previamente definidas.

- Agrupamiento (*Clustering*): Es una tarea similar a la clasificación pero no existen clases predefinidas. El proceso automáticamente determinará cuáles son las particiones.

- Sumarización: Área que entiende sobre técnicas de extracción de aquellas partes (palabras, frases, oraciones, párrafos) que contienen la semántica que determina la esencia de un documento.

- Detección de novedades (*Novelty Detection*): Se basa en la determinación de la introducción de nuevos tópicos o temas a un SRI.

[2] http://www.altavista.com/
[3] http://www.google.com/

- Respuestas a Preguntas (*Question Answering*): Consiste en hallar aquellas porciones de texto de un documento que satisfacen expresamente a una consulta, es decir, la respuesta concreta a una pregunta dada.

- Extracción de Información: Extraer aquellas porciones de texto con una alta carga semántica y establecer relaciones entre los términos o pasajes extraídos.

- Recuperación *cross-language*: Hallar documentos escritos en cualquier lenguaje que son relevantes a una consulta expresada en otro lenguaje (búsqueda multilingual).

- Búsquedas Web: Se refiere a los SRI que operan sobre un corpus web privado (intranet) o público (Internet). La web ha planteado nuevos desafíos al área de RI, debido a sus características particulares como – por ejemplo – dinámica y tamaño.

- Recuperación de Información Distribuida: A diferencia de los SRI clásicos donde el corpus y las estructuras de datos que auxilian a la búsqueda están centralizadas, aquí se plantea la tarea sobre los mismos elementos pero distribuidos sobre una red de computadoras.

- Modelado de Usuarios: Esta área – a partir de la interacción de los usuarios con un SRI – estudia como se generan de forma automática perfiles que definan las necesidades de información de éstos.

- Recuperación de Información Multimedia: Más allá de que los SRI tradicionales operan sobre corpus de documentos textuales, la recuperación de información tiene que tratar con otras formas alternativas de representación como imágenes, registro de conversaciones y video.

- Desarrollo de Conjuntos (data-sets) de Prueba: A los efectos de evaluar SRI completos o nuevos métodos y técnicas es necesario disponer de juegos de prueba normalizados (corpus con preguntas y respuestas predefinidas, corpus clasificados, etc.). Esta área tiene que ver con la producción tales conjuntos, a partir de diferentes estrategias que permitan reducir la complejidad de la tarea, manejando la dificultad inherente a la carga de subjetividad existente.

1.2 – ¿Recuperación de Información o Recuperación de Datos?

Muchos usuarios se encuentran familiarizados con el concepto de recuperación de datos (RD), especialmente aquellos que – a menudo – interactuan con sistemas de consulta en bases de datos relacionales ó en registros de alguna naturaleza, como por ejemplo, un registro de los empleados de una organización. Sin embargo, hay diferencias significativas en los conceptos que definen que el tratamiento de las unidades (datos o información) en cada caso sean completamente diferentes.

Básicamente, existen diferencias sustanciales en cuanto a los objetos con que se trata y su representación, la especificación de las consultas y los resultados.

En el área de RD los objetos que se tratan son estructuras de datos conocidas. Su representación se basa en un formato previo definido y con un significado implícito (hay una sintaxis y semántica no ambigua) para cada elemento. Por ejemplo, una tabla en una base de datos que almacena instancias de clientes de una organización posee un conjunto de columnas que definen los atributos de todos los clientes y cada fila corresponde a uno en particular. Nótese que cada elemento (atributo) tiene un dominio conocido y su semántica está claramente establecida. Por otro lado, en el área de RI la unidad u objeto de tratamiento es básicamente un documento de texto – en general – sin estructura.

En cuanto a la especificación de las consultas, en el área de RD se cuenta con una estructura bien definida dada por un lenguaje de consulta que permite su especificación de manera exacta. Las consultas no son ambiguas y consisten en un conjunto de condiciones que deben cumplir los ítems a evaluar para que la misma se satisfaga. Por ejemplo, en el modelo de bases de datos, las consultas especifican – entre otros – utilizando el lenguaje SQL (Structured Query Language) cuya semántica es precisa:

SQL	**En lenguaje natural**
SELECT *	Seleccionar todos los clientes de
FROM Clientes	Chivilcoy que deban más de 10000 pesos
WHERE Localidad = “Chivilcoy”	(se sabe, por definición, que lo que deben
AND Saldo_Cuenta > 10000	es su saldo de cuenta)

En este ejemplo, se puede ver la clara semántica de la consulta en SQL a partir de que se conoce que existe un atributo Localidad y otro Saldo_Cuenta y lo que cada uno representa. Sin embargo, esto no es tan directo ni tan simple cuando se trata de recuperar documentos en el contexto de la RI. En primer lugar, debido a que la necesidad de un usuario puede ser difícil de expresar. Por ejemplo, supóngase que se desea encontrar:

“*Documentos que contengan información biográfica de los entrenadores de los equipos de fútbol de Argentina que ganaron más torneos en los últimos 10 años*”

La primera dificultad consiste en construir una expresión de consulta que refleje exactamente esta necesidad de información del usuario. Especialmente, si se tiene en cuenta que para resolverla completamente quizá primero se requiera de conocer información parcial, por ejemplo, “ganaron más torneos en los últimos 10 años”. ¿Qué significa “ganaron más torneos”? Esta es una situación subjetiva y – en muchos casos – el sistema debe manejar estas cuestiones, junto con ambigüedades (por ejemplo, palabras cuyo significado está determinado por el contexto) e incompletitud de la mejor manera posible. De hecho, los documentos y las expresiones de consulta se interpretan de forma que el proceso de recuperación determine un grado de similitud entre éstos.

Finalmente, en un sistema de RD los resultados consisten en el conjunto completo de elementos que satisfacen todas las condiciones del *query*. Como la consulta no admite errores, el resultado es exacto, ni uno más, ni uno menos. Y el orden de aparición es simplemente casual (a menos que específicamente se desee ordenar por alguna columna), pero en todos los casos este orden es irrelevante respecto de la consulta y no significa nada, es decir, no se puede implementar sistema de ranking alguno. En el área de RI, aparece el concepto de relevancia y la salida (respuesta) se encuentra

confeccionada de acuerdo con algún criterio que evalúa la “similitud” que existe entre la consulta y cada documento. Por lo tanto, el resultado es un ranking (que no es sinónimo de “orden”, tal como se lo entiende habitualmente en RD), donde la primera posición corresponde al documento más relevante a la consulta y así decrece sucesivamente. El proceso de recuperación de información puede retornar documentos que no sean relevantes para el usuario, es decir, que el conjunto de respuesta no es exacto.

A continuación, se resumen las diferencias más significativas entre un SRI y un sistema de RD como lo es un Sistema de Gestión de Bases de Datos (SGBD).

	SGBD	**SRI**
Estructura	Información estructurada con semántica bien definida.	Información semi o no estructurada.
Recuperación	Determinística. Todo el conjunto solución es relevante para el usuario	Probabilística. Una porción de los documentos recuperados puede no ser relevante.
Consulta y Lenguaje	Especificación precisa (no hay ambigüedad). Lenguaje formal, preciso y estructurado.	Hay imprecisión en su formulación. Lenguaje natural, ambiguo y no estructurado.
Resultados	Aciertos exactos	Aciertos parciales

En el mismo sentido, Blair [3] [4] comparó la RD (a) con la RI (b) tomando trece parámetros, definidos por características de los sistemas y criterios de evaluación.

1) **Búsqueda**
   a) Es directa ("Quiero saber X"). La respuesta correcta está o no.
   b) Es indirecta ("Quiero saber sobre X"). Una respuesta puede no ser la correcta.

2) **Representación de la necesidad de información**
   a) El planteamiento formal de la consulta y las necesidades de información del usuario son similares (datos estructurados y lenguaje de consulta estructurado).
   b) No siempre se puede asegurar que la consulta represente realmente las necesidades de información del usuario.

3) **Criterios de éxito**
   a) Corrección: Los sistemas de recuperación de datos recuperan la respuesta correcta. No hay ambigüedad en el resultado.
   b) Relevancia: No hay respuestas correctas sino documentos relevantes.

4) **Velocidad de respuesta**
   a) Es predecible y depende – principalmente – del acceso físico debido a que la tarea principal de la recuperación consiste en “navegar” sobre una estructura de datos.

b) No es predecible debido a que el proceso completo puede constar de varias etapas de retroalimentación hasta encontrar un conjunto de documentos relevantes que satisfaga una demanda de información particular.

5) Representación de la información

a) Las formas de representar los datos son finitas. Existen pocas formas de poner – por ejemplo – una fecha de nacimiento.
b) Las formas de representar los documentos son virtualmente infinitas, por la ambigüedad del lenguaje.

6) Conjunto de respuestas posibles

a) Al no haber formas distintas de representar los datos, el número de posibles respuestas alternativas es limitado.
b) Formas alternativas de representar los documentos implican demasiadas preguntas posibles por documento.

7) Ausencia de respuesta o respuestas no útiles

a) Se deriva de que los datos no están en la base de datos.
b) El documento puede existir pero no se ha expresado una pregunta en la pregunta adecuada para poder recuperarlo.

8) Tipos de búsquedas

a) Búsqueda exacta.
b) A) simple ("deme algunos documentos sobre X"), B) exhaustiva (“deme todos los documentos sobre X"), y C) de existencia ("existen documentos sobre X?").

9) Escalabilidad del sistema

a) Tiene poco impacto en el funcionamiento de la recuperación.
b) Puede tener un impacto negativo. Hacer un sistema escalable puede exigir modificar las estrategias de búsqueda.

10) Tipos de sistemas

a) En recuperación de datos son orientados a transacciones.
b) En base al soporte de decisiones (por ejemplo, gestión del conocimiento, investigación)

11) Delegación de la búsqueda

a) Es factible delegarla.
b) Es difícil delegar una búsqueda por la propia naturaleza ambigua del lenguaje.

12) Justificación económica

a) Se puede calcular con cierta exactitud la amortización.
b) Todavía es difícil su cálculo.

13) Contexto

a) Por la propia estructura formal del modelo, el contexto es preciso y sin ambigüedad.

b) En las colecciones de documentos pueden existir varios contextos. De aquí surge la necesidad de operar con técnicas de desambiguación para aumentar la precisión del conjunto solución a una consulta.

Otros autores también establecieron las diferencias entre ambos conceptos: Grossman y otro [24] claramente muestran la diferencia cuando enuncian que "*la recuperación de información es encontrar documentos relevantes, no encontrar simples correspondencias a unos patrones de bits*". Nótese la diferencia sustancial que existe en tratar de encontrar documentos "relevantes" a una consulta o – simplemente – encontrar aquellos donde "coinciden" patrones de términos o se cumplen ciertas condiciones. En el caso de la RD, la tarea es relativamente sencilla, mientras que en área de RI es extremadamente compleja y no existe aún una solución definitiva al problema.

1.3 – La interacción del usuario con el SRI

La tarea de recuperar información puede ser planteada de diversas formas, de acuerdo a cómo el usuario interactúa con el sistema o bien qué facilidades éste le brinda. Básicamente, la tarea se la puede dividir en:

1) **Recuperación inmediata**: El usuario plantea su necesidad de información y – a continuación – obtiene referencias a los documentos que el sistema evalúa como relevantes. Existen dos modalidades:

   a) **Búsqueda (propiamente dicha) o recuperación "*ad-hoc*"**, donde el usuario formula una consulta en un lenguaje y el sistema la evalúa y responde. En este caso, el usuario tiene suficiente comprensión de su necesidad y sabe cómo expresar una consulta al sistema. Un ejemplo clásico son los buscadores de Internet como Google[4], Yahoo[5] o MSN Search[6].

   b) **Navegación o *browsing***: En este caso, el usuario utiliza un enfoque diferente al anterior. El sistema ofrece una interfase con temas donde el usuario "navega" por dicha estructura y obtiene referencias a documentos a relacionados. Esto facilita la búsqueda a usuarios que no pueden definir claramente cómo comenzar con su consulta e – inclusive – van definiendo su necesidad a medida que observan diferentes documentos. Es este enfoque no se formula consulta explícita. Un ejemplo típico es el proyecto Open Directory[7].

   En ambos casos, la colección es relativamente estática, es decir, se parte de un conjunto de documentos y la aparición de nuevos no es muy significativa. Por otro lado, las consultas son las que se van modificando ya que este proceso es proactivo por parte del usuario.

[4] http://www.google.com/
[5] http://www.yahoo.com/
[6] http://www.msn.com/
[7] http://www.dmoz.org/

2) **Recuperación diferida**: El usuario especifica sus necesidades y el sistema entregará de forma continua los nuevos documentos que le lleguen y concuerden con ésta. Esta modalidad recibe el nombre de **filtrado y ruteo** y la necesidad del usuario – generalmente – define un "perfil" (*profile*) de los documentos buscados. Nótese que un "perfil" es – de alguna forma – un *query* y puede ser tratado como tal. Cada vez que un nuevo documento arriba al sistema se compara con el perfil y – si es relevante – se envía al usuario. Un ejemplo, es el servicio provisto por la empresa Indigo Stream Technologies denominado GoogleAlert[8].

   En esta modalidad la consulta es relativamente estática (corresponde al *profile*) y el usuario tiene un rol pasivo. La dinámica está dado por la aparición de nuevos documentos y es lo que determina mas resultados para el usuario.

En algunos casos, se plantea que documentos y consultas son objetos de la misma clase por lo que estos enfoques son – de alguna manera – visiones diferentes de una misma problemática. Bajo este punto de vista, documentos y consultas se pueden intercambiar. Sin embargo, esto no es siempre posible debido al tratamiento que se aplica a cada uno en diferentes sistemas. Algunos sistemas representan *queries* y documentos de diferente manera. Es más, existe una diferencia obvia en cuanto a la longitud de uno y otro que se tiene en cuenta bajo ciertos modelos de recuperación. Finalmente, en un sistema de búsqueda, existe el concepto de ranking de los documentos respuesta, mientras que en un modelo de filtrado, cada nuevo objeto es relevante a un perfil o no.

1.4 – El concepto de relevancia

Como mencionamos, la recuperación de información intenta resolver el problema de encontrar documentos relevantes que satisfagan la necesidad de información de un usuario. Sin embargo, se ha planteado la dificultad para llevar a cabo esta tarea debido a la imposibilidad de expresar exactamente tal necesidad. Además, la noción de relevancia es un juicio subjetivo [61] y depende de diferentes factores relacionados más cercanamente con el usuario. La relevancia de un documento respecto de un *query* se refiere a cuánto el primero responde al segundo. De igual manera, luego el usuario evalúa cuánto, es decir, en qué medida, se satisface su necesidad de información [25].

Es por ello, que se plantea la relevancia como similitud, de manera de poder comparar documentos con consultas y – bajo ciertos criterios – definir una medida de distancia entre ambos. Por lo tanto, se puede plantear la idea de que "un documento es relevante a una consulta si son similares", donde la medida de similitud puede estar basada en diferentes criterios (coincidencias de términos, significado de éstos, frecuencia de aparición de términos y distribución del vocabulario, entre otros).

Martínez Méndez y otros [36] resaltan la dificultad para determinar la relevancia o no de un documento respecto de una consulta. Plantean – por ejemplo – que dos personas pueden juzgar un mismo documento de diferente manera y que es difícil establecer los criterios para la evaluación de la relevancia. Finalmente, mencionan la idea de relevancia parcial, es decir, cuando solo una parte del documento se considera relevante.

[8] http://www.googlealert.com/

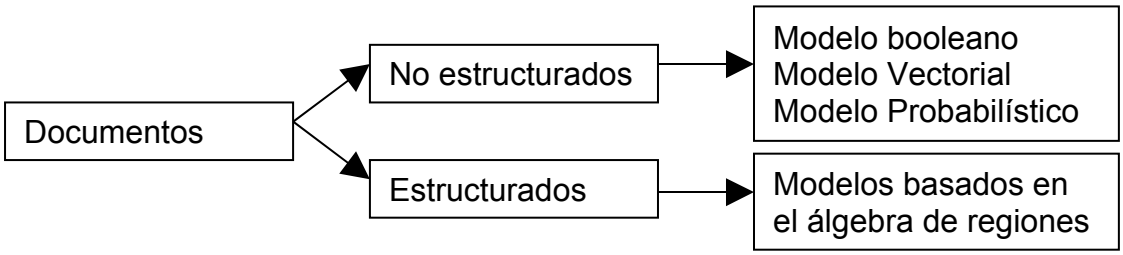
Por otro lado, como el *query* no describe exactamente la necesidad de información del usuario, algunos autores [25] definen el concepto de "pertinencia", donde se incluyen las restricciones impuestas por el SRI. Este concepto está relacionado con la utilidad del documento para el usuario [36], de acuerdo a la necesidad de información original que guió su búsqueda, independientemente si es en parte o todo el documento.

Sin embargo – y a pesar de las dificultades para determinarla – el concepto genérico de relevancia es aceptado ampliamente por la comunidad de RI para evaluar la respuesta de un SRI respecto de una consulta de un usuario, la cual – como ya mencionamos – surge a partir de una necesidad de información.

1.5 – Modelos de RI

Los SRI toman un conjunto de documentos (colección) para procesar y luego poder responder consultas. De forma básica, podemos clasificar los documentos en estructurados y no estructurados. Los primeros son aquellos en los que se pueden reconocer elementos estructurales con una semántica bien definida, mientras que los segundos corresponden a texto libre, sin formato. La diferencia fundamental de un SRI que procese documentos estructurados se encuentra en que puede extraer información adicional al contenido textual, la cual utiliza en la etapa de recuperación para facilitar la tarea y aumentar las prestaciones.

A partir de lo expresado anteriormente se presenta una posible clasificación de modelos de RI – la cual no es exhaustiva – de acuerdo a características estructurales de los documentos.



A continuación se describen – de forma somera – los modelos clásicos y el álgebra de regiones. Más adelante, en el capítulo 4, se profundizará en los modelos booleano y vectorial.

1.5.1 – Modelo booleano

En el modelo booleano la representación de la colección de documentos se realiza sobre una matriz binaria documento–término, donde los términos han sido extraídos manualmente o automáticamente de los documentos y representan el contenido de los mismos.

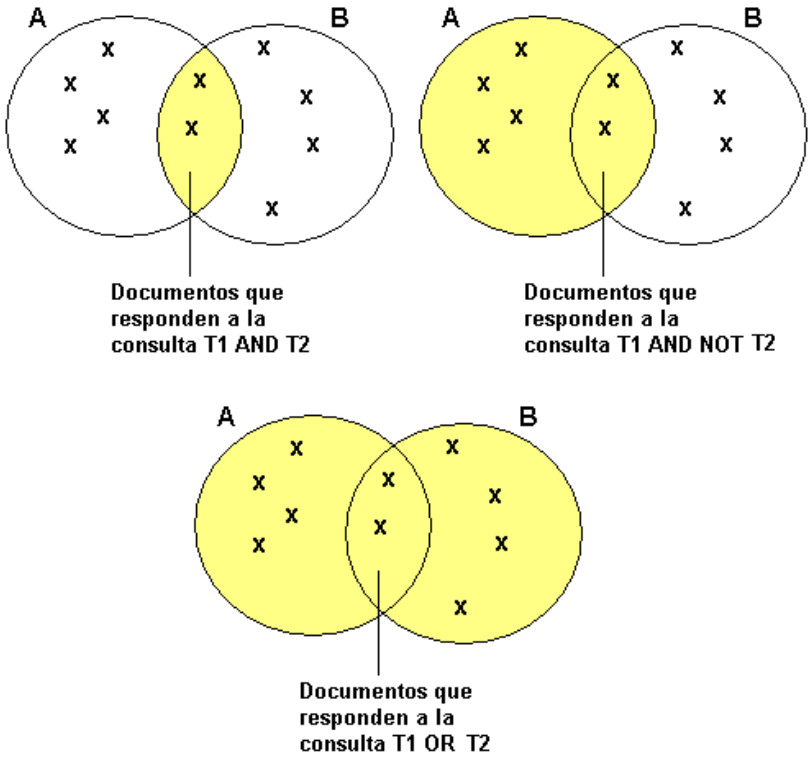
	t1	**t2**	**t3**	**…**	**tn**
d1	1	0	0		

d2	1	1	1		
d3	0	0	1		
…					
dn					

Matriz binaria documento – término

Las consultas se arman con términos vinculados por operadores lógicos (AND, OR, NOT) y los resultados son referencias a documentos donde cuya representación satisface las restricciones lógicas de la expresión de búsqueda. En el modelo original no hay ranking de relevancia sobre el conjunto de respuestas a una consulta, todos los documentos poseen la misma relevancia. A continuación se muestra un ejemplo mediante conjuntos de operaciones utilizando el modelo booleano.

A = {Documentos que contienen el término T1}
B = {Documentos que contienen el término T2}



Si bien es el primer modelo desarrollado y aún se lo utiliza, no es el preferido por los ingenieros de software para sus desarrollos. Existen diversos puntos en contra que hacen que cada día se lo utilice menos y – además – se han desarrollado algunas extensiones, bajo el nombre modelo booleano extendido [64] [51], que tratan de mejorar algunos puntos débiles.

1.5.2 – Modelo Vectorial

Este modelo fue planteado y desarrollado por Gerard Salton [49] y – originalmente – se implementó en un SRI llamado SMART. Aunque el modelo posee más de treinta años, actualmente se sigue utilizando debido a su buena performance en la recuperación de documentos.

Conceptualmente, este modelo utiliza una matriz documento–término que contiene el vocabulario de la colección de referencia y los documentos existentes. En la intersección de un término *t* y un documento *d* se almacena un valor numérico de importancia del término *t* en el documento *d*; tal valor representa su *poder de discriminación*. Así, cada documento puede ser visto como un vector que pertenece a un espacio n-dimensional, donde *n* es la cantidad de términos que componen el vocabulario de la colección. En teoría, los documentos que contengan términos similares estarán a muy poca distancia entre sí sobre tal espacio. De igual forma se trata a la consulta, es un documento más y se la mapea sobre el espacio de documentos. Luego, a partir de una consulta dada es posible devolver una lista de documentos ordenados por distancia (los más relevantes primero). Para calcular la semejanza entre el vector consulta y los vectores que representan los documentos se utilizan diferentes fórmulas de distancia, siendo la más común la del coseno.

Obsérvese el siguiente ejemplo donde se representa a un documento d y a una consulta c:

Documento: "*La República Argentina ha sido nominada para la realización del X Congreso Americano de Epidemiología en Zonas de Desastre. El evento se realizará ...*"

Consulta: "*argentina congreso epidemiología*"

	argentina	**...**	**congreso**	**epidemiología**	**...**
d_1	0.5		0.3	0.2	
...					
d_n					
Consulta	0.4		0.3	0.3	

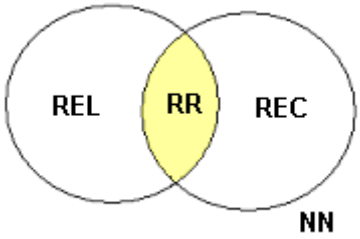
Matriz término-documento con pesos normalizados entre 0 y 1

Representación gráfica del documento y la consulta

1.5.3 – Modelo probabilístico

Fue propuesto por Robertson y Spark-Jones [47] con el objetivo de representar el proceso de recuperación de información desde el punto de vista de las probabilidades. A partir de una expresión de consulta se puede dividir una colección de N documentos en cuatro subconjuntos distintos: REL conjunto de documentos relevantes, REC conjunto de

documentos recuperados, RR conjunto de documentos relevantes recuperados y NN el conjunto de documentos no relevantes no recuperados.



El resultado ideal de a una consulta se da cuando el conjunto REL es igual REC. Como resulta difícil lograrlo en primera intención, el usuario genera una descripción probabilística del conjunto REL y a través de sucesivas interacciones con el SRI se trata de mejorar la performance de recuperación. Dado que una recuperación no es inmediata dado que involucra varias interacciones con el usuario y que estudios han demostrado que su performance es inferior al modelo vectorial, su uso es bastante limitado.

1.5.4 – Modelos para documentos estructurados

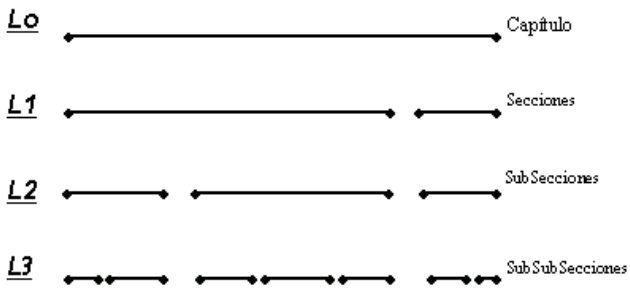
Los modelos clásicos responden a consultas, buscando sobre una estructura de datos que representa el contenido de los documentos de una colección, únicamente como listas de términos significativos. Un modelo de recuperación de documentos estructurados utiliza la estructura de los mismos a los efectos de mejorar la performance y brindar servicios alternativos al usuario (por ejemplo, uso de memoria visual, recuperación de elementos multimedia, mayor precisión sobre el ámbito de la consulta y demás).

La estructura de los documentos a indexar está dada por marcas o etiquetas, siendo los estándares más utilizados el SGML (*Standard General Markup Language*), el HTML *(HyperText Markup Language*), el PDF (*Portable Document Format*), el XML (*eXtensible Markup Language*) y LATEX.

Al poseer la descripción de parte de la estructura de un documento es posible generar un grafo sobre el que se navegue y se respondan consultas de distinto tipo, por ejemplo:

- Por estructura: *¿Cuáles son las secciones del segundo capítulo?*
- Por metadatos o campos: *Documentos de "Editorial UNLu" editados en 1998*
- Por contenido: *Término "agua" en títulos de secciones*
- Por elementos multimedia: *Imágenes cercanas a párrafos que contengan Bush*

Para Baeza-Yates existen dos modelos en esta categoría "nodos proximales" [37] y "listas no superpuestas" [8]. Ambos modelos se basan en almacenar las ocurrencias de los términos a indexar en estructuras de datos diferentes, según aparezcan en algún elemento de estructura (región) o en otro como capítulos, secciones, subsecciones y demás. En general, las regiones de una misma estructura de datos no poseen superposición, pero regiones en diferentes estructuras sí se pueden superponer.



Nótese que en el ejemplo anterior existen cuatro niveles de listas para almacenar información relativa a contenido textual y estructura de un documento tipo libro. En general, los tipos de consultas soportados son simples:

- Seleccione una región que contenga una palabra dada
- Seleccione una región X que no que no contenga una región Y
- Seleccione una región contenida en otra región

Sobre el ejemplo de estructura planteado un ejemplo de consulta sería:

[subsección[+] CONTIENE "'tambo"]

Como respuesta el SRI buscaría subsecciones y sub-subsecciones que contengan el término "tambo".

Cabe mencionar que algunos motores de búsqueda de Internet ya utilizan ciertos elementos de la estructura de un documento – por ejemplo, los títulos – a los efectos de realizar tareas de ranking, resumen automático, clasificación y otras.

La expansión de estos lenguajes de demarcación, especialmente en servicios sobre Internet, hacen que se generen y publiquen cada vez más documentos semiestructurados. Es necesario – entonces – desarrollar técnicas que aprovechen el valor agregado de los nuevos documentos. Si bien – en la actualidad – éstas no se encuentran tan desarrolladas como los modelos tradicionales, consideramos su evolución como una cuestión importante en el área de RI, especialmente a partir de investigaciones con enfoques diferentes que abordan la problemática [18] [38] [45].

1.6 – La RI en la era de la Web

Con la aparición de la web surgieron nuevos desafíos para resolver en el área de recuperación de información debido – principalmente – a sus características y su tamaño. La web puede ser vista como un gran repositorio de información, completamente distribuido sobre Internet y accesible por gran cantidad de usuarios. Por sus orígenes como un espacio público existen millones de organizaciones y usuarios particulares que incorporan, quitan ó modifican contenido continuamente, por lo que su estructura no es estática.

Su contenido no respeta estándares de calidad, ni estilos ni organización. Como medio de publicación de información de naturaleza diversa se ha convertido en un servicio

de permanente crecimiento. Una de las características de la información publicada en la web es su dinámica, dado que pueden variar en el tiempo tanto los contenidos como su ubicación [6] [33].

El tamaño de la web es imposible de medir exactamente y muy difícil de estimar. Sin embargo, se calcula que son decenas de exabytes de información, y crece permanentemente. Está formada por documentos de diferente naturaleza y formato, desde páginas HTML hasta archivos de imágenes pasando por gran cantidad de formatos estándar y propietarios, no solamente con contenido textual, sino también con contenido multimedial.

La búsqueda de información en la web es una práctica común para los usuarios de Internet y los sistemas de recuperación de información web (conocidos como motores de búsqueda) se han convertido en herramientas indispensables para los usuarios. Su arquitectura y modo de operación se basan en poder recolectar mediante un mecanismo adecuado los documentos existentes en los sitios web. Una vez obtenidos, se llevan a cabo tareas de procesamiento que permiten extraer términos significativos contenidos dentro de los mismos, junto con otra información, a los efectos de construir estructuras de datos (índices) que permitan realizar búsquedas de manera eficiente. Luego, a partir de una consulta realizada por un usuario, un motor de búsqueda extraerá de los índices las referencias que satisfagan la consulta y se retornará una respuesta rankeada por diversos criterios al usuario. El modo de funcionamiento de los diferentes motores de búsqueda puede diferir en diversas implementaciones de los mecanismos de recolección de datos, los métodos de indexación y los algoritmos de búsqueda y ranking.

Sin embargo, esta tarea no es sencilla y se ha convertido en un desafío para los SRI debido las características propias de la web. Baeza-Yates [2] plantea que hay desafíos de dos tipos:

a) Respecto de los datos

- Distribuidos: La web es un sistema distribuido, donde cada proveedor de información publica su información en computadoras pertenecientes a redes conectadas a Internet, sin una estructura ó topología predefinida.

- Volátiles: La dinámica del sistema hace que exista información nueva a cada momento ó bien que cambie su contenido ó inclusive desaparezca otra que se encontraba disponible.

- No estructurados y redundantes: Básicamente, la web está formada de páginas HTML, las cuales no cuentan con una estructura única ni fija. Además, mucho del contenido se encuentra duplicado (por ejemplo, espejado).

- Calidad: En general, la calidad de la información publicada en la web es altamente variable, tanto en escritura como en actualización (existe información que puede considerarse obsoleta), e inclusive existe información con errores sintácticos, ortográficos y demás.

- Heterogeneidad: La información se puede encontrar publicada en diferentes tipos de medios (texto, audio, gráficos) con diferentes formatos para cada uno de éstos. Además, hay que contemplar los diferentes idiomas y diferentes alfabetos (por ejemplo, árabe ó chino).

b) Respecto de los usuarios.

- Especificación de la consulta: Los usuarios encuentran dificultades para precisar – en el lenguaje de consulta – su necesidad de información.

- Manejo de las respuestas: Cuando un usuario realiza una consulta se ve sobrecargado de respuestas, siendo una parte irrelevante.

Estas características – sumadas al tamaño de la web – imponen restricciones a las herramientas de búsqueda en cuanto a la cobertura y acceso a los documentos, exigiendo cada vez mayores recursos computacionales (espacio de almacenamiento, ancho de banda de las redes, ciclos de CPU) y diferentes estrategias para mejorar la calidad de las respuestas.

1.7 – Actividades

1) Responda las siguientes preguntas sin utilizar recursos digitales. Describa detalladamente el proceso utilizado en cada caso. Trate de valorar el costo asociado y recursos utilizados a cada búsqueda mediante parámetros propios.
   a) ¿Cómo está formada la molécula de agua?
   b) ¿Quién era Karol Wojtyla?
   c) ¿Dónde vive Julio Bocca?
   d) ¿Cuándo finalizó la guerra del golfo?
   e) ¿Por qué los días se acortan en invierno?

   Suponga que usted no sabe la respuesta a ninguna de las preguntas.

2) Utilizando como recurso el motor de búsqueda Google trate de satisfacer las siguientes necesidades de información. Indique en cada caso las consultas que tuvo que realizar para llegar al objetivo (si llegó), cuántas referencias de la respuesta revisó en cada caso y en cuál encontro lo buscado (si es necesario, puede escribir las consultas en inglés). Suponga que usted no sabe la respuesta a ninguna de las preguntas.
   a) Se requiere saber cuáles tumbas de faraones egipcios fueron halladas por arqueólogos europeos.
   b) ¿Por qué se hundió el Titanic?
   c) ¿Cómo se llama el disco de Pink Floyd editado luego de la caída del muro de Berlín.

3) Ejecute las siguientes consultas en los motores de búsqueda Google, Teoma (http://www.teoma.com) y Vivisimo (http://www.vivisimo.com). ¿Qué diferencias encuentra en cada uno respecto de la respuesta? ¿Qué ventajas y desventajas visibles encuentra? Recupere los primeros 10 resultados de la búsqueda e indique si para usted son relevantes a la consulta (si es necesario, puede escribir las consultas en inglés).
   a) ¿Qué sistemas compañero a compañero sirven para intercambiar archivos de música?
   b) Se requiere conocer la nómina de los presidentes Argentinos desde Rivadavia a la actualidad.
   c) ¿Cuáles son los torneos del circuito de tenis denominados Grand Slam?

4) Utilizando como recurso Open Directory (http://www.dmoz.org) indique cuántas categorías existen relacionadas con seguridad en computadoras. ¿Cómo llegó al resultado? ¿Qué información le aporta el servicio de directorio? ¿Con qué servicio es comparable en cuanto a funcionalidad?

5) Ingrese en GoogleAlert (http://www.googlealert.com) y explique en qué consiste y qué servicios ofrece. Genere una búsqueda y realice el monitoreo durante 1 semana. Trate de especificar tres situaciones concretas en las cuales este servicio resulta altamente útil. Indique posibles mejoras que le haría para situaciones particulares.

6) Evalúe el servicio de “Respuestas a Preguntas” AskJeeves (http://www.askjeeves.com) y responda:

a) Según su criterio ¿En qué situaciones lo considera un recurso útil?
b) ¿Qué inconvenientes o restricciones encuentra en éste?

Para sus pruebas utilice consultas tanto en español como en inglés. Además, evalúelo con preguntas de conocimiento general (¿Dónde vive Madonna?) y de alcance local (¿Quién es el actual arquero de River Plate?).

Capítulo 2

Evaluación de los sistemas de recuperación de información

Como hemos mencionado en el capítulo anterior, existen diferentes aproximaciones para resolver el problema central del área de RI. De aquí que contamos con sistemas basados en diferentes modelos y algoritmos, los cuales – como todo sistema – deben poder ser evaluados bajo ciertos criterios. Esta tarea permite medir los parámetros de funcionamiento que valoran al sistema y – además – posibilitan la comparación entre distintos SRI.

En sistemas tradicionales – como los de recuperación de datos – los parámetros típicos de evaluación son *tiempo* y *espacio*. La medición del tiempo de respuesta y de la cantidad de espacio de almacenamiento que el sistema utiliza brindan parámetros concretos y relativamente fáciles de medir. Mientras un sistema responda más velozmente y requiera menos almacenamiento, mejor será. El balance entre estos dos parámetros depende de los objetivos de diseño del sistema. Por ejemplo, si el requerimiento mayor corresponde a la velocidad del sistema para entregar respuestas porque con éstas se toman decisiones en tiempo real, no importará demasiado la cantidad de estructuras de datos que se van a implementar y que permitirán acelerar las búsquedas. Por el contrario, si el sistema va a correr sobre una PDA, el espacio de almacenamiento es muy importante y – tal vez – se pueda "esperar" un poco más por la respuesta.

Sin embargo, la evaluación de un sistema de RI no es una tarea sencilla. Debido a que el conjunto de respuesta no es exacto se requiere ponderar cómo éste se ajusta a la consulta y – peor aún – ésta a la necesidad de información del usuario. Aquí aparecen las cuestiones subjetivas que se plantean al especificar un *query*, al adoptar una representación lógica de los documentos de la colección y al utilizar una función de ranking determinada.

Por lo tanto, la evaluación más común se orienta a determinar cuán preciso es el conjunto respuesta a partir del concepto de relevancia (planteado en el Capítulo 1) de cada documento respecto de la consulta. Para Baeza-Yates [2] este tipo de evaluación corresponde a la *evaluación de la performance de la recuperación*. En el mismo sentido, van Risbergen [van Risbergen, 1999] plantea medir la *efectividad del sistema de recuperación*, la cual cuantifica su capacidad para recuperar documentos relevantes mientras no recupera documentos no relevantes. Un sistema más efectivo permite satisfacer en mayor medida la necesidad de un usuario.

2.1 – Medidas de Evaluación

Desde los primeros esfuerzos relacionados con la evaluación de los SRI hasta la actualidad, la aproximación clásica para describir la *performance de la recuperación*

consiste en determinar **cuántos documentos relevantes se recuperaron y cómo se rankearon para entregarlos al usuario**.

Cuando un usuario plantea un *query* a un SRI, obtiene como respuesta una lista de documentos de acuerdo a un ranking determinado por el sistema. Como ya hemos mencionado, dicha respuesta está formada por documentos relevantes y otros que no lo son y la lista – generalmente – no contiene todos los documentos de la colección (en colecciones grandes, sería imposible de revisar toda la respuesta). En el gráfico 1 se muestra esta situación. Dada una consulta cualquiera, un SRI recuperará el grupo identificado como C, de los cuales solo una parte son relevantes (D).

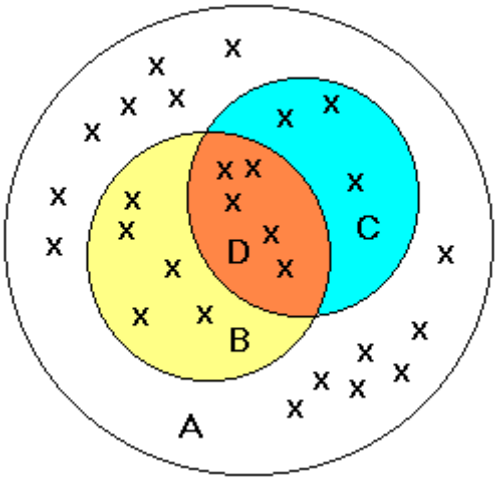


Gráfico 1 – Documentos en una colección

A: Todos – **B**: Relevantes – **C**: Recuperados – **D**: Relevantes recuperados (B ∩ C)

A partir de esta situación, se puede establecer la siguiente tabla de contingencia para la evaluación.

	Recuperados	**No Recuperados**	
Relevantes	Rel-Rec (**w**) Positivos verdaderos	Rel-NoRec (**x**) Falsos positivos	Total de docs Relevantes: **w + x**
No Relevantes	NoRel-Rec (**y**) Falsos negativos	NoRel-NoRec (**z**) Negativos verdaderos	Total de docs no Relevantes: **y + z**
	Total de docs Recuperados: **w + y**	Total de docs no Recuperados: **x + z**	Total docs: **w + x + y +z**

Existen dos medidas ampliamente aceptadas en la comunidad de RI denominadas Precisión (en inglés, *Precision*) y Exhaustividad (*Recall*) planteadas por Cleverdon [13] hace varios años. La **Exhaustividad** se define como la proporción de los documentos

relevantes que han sido recuperados y permite evaluar la habilidad del sistema para encontrar todos los documentos relevantes de la colección. Tal concepto podría resumirse a partir de la siguiente cuestión "¿Son todos los que están o nos faltan algunos?".

$$Exhaustividad(E) = \frac{Cantidad\_documentos\_relevantes\_recuperados}{Cantidad\_documentos\_relevantes} \qquad E = \frac{w}{w + x}$$

La **Precisión** se define como la proporción de los documentos recuperados que son relevantes y permite evaluar la habilidad del sistema para rankear primero la mayoría de los documentos relevantes. Expresado en forma de pregunta: ¿Son todos relevantes o se "filtraron" algunos que no lo son?

$$\Pr ecisión(P) = \frac{Cantidad\_documentos\_relevantes\_recuperados}{Cantidad\_documentos\_recuperados} \qquad P = \frac{w}{w + y}$$

Estas dos medidas se encuentran altamente relacionadas. Empíricamente se ha comprobado que una alta exhaustividad se acompaña de una muy baja precisión y viceversa (Gráfico 2), es decir, existe una relación inversa entre las [14].

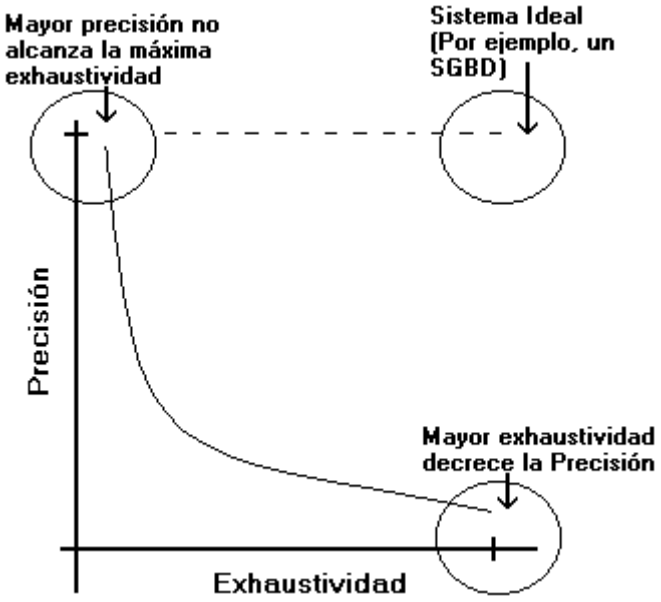


Gráfico 2 – Relación entre Precisión y Exhaustividad

Existe un compromiso entre Exhaustividad y Precisión, es decir, al aumentar la Exhaustividad recuperando mayor cantidad de documentos, veremos disminuir la Precisión. Esto se explica en el hecho – ya mencionado – que la salida de un SRI es un conjunto aproximado (no exacto) y – por lo tanto – entre ésta se encontrarán documentos no relevantes. Por el contrario, si recuperamos unos pocos documentos y todos son relevantes se tendrá una precisión máxima, pero seguramente se están perdiendo documentos útiles por no ser recuperados. El sistema ideal es aquel que siempre recupera todos los documentos relevantes y solo esos, situación que – hasta el momento – no existe.

Para analizar estas medidas planteamos el siguiente ejemplo: Existe una colección *D* la cual posee 100 documentos, digamos:

$D = \{d_1, d_2, d_3, d_4, d_5, \ldots, d_{98}, d_{99}, d_{100}\}$

Ahora, supóngase que para una consulta *q* existen 10 documentos relevantes, *R*:

$R = \{d_2, d_{45}, d_{70}, d_{77}, d_{79}, d_{81}, d_{82}, d_{88}, d_{90}, d_{91}\}$

A pedido del usuario, el sistema entregó los primeros 12 documentos, *A*, rankeados de la siguiente forma:

$A = \{\mathbf{d_{79}}, d_{10}, \mathbf{d_{90}}, d_{13}, d_{20}, \mathbf{d_{45}}, d_{60}, d_{30}, \mathbf{d_{77}}, d_{21}, \mathbf{d_{88}}, d_{100}\}$

(se han resaltado los documentos pertenecientes a *R,* es decir, los relevantes)

Los cálculos de *performance de la recuperación* para un tamaño de respuesta de 12 documentos resultan:

P = 5 / 12 = 0.42
E = 5 / 10 = 0.50

Sin embargo, se puede modificar el tamaño de la respuesta a los efectos de intentar recuperar más documentos relevantes, es decir, para aumentar la exhaustividad. En tal caso, supóngase una nueva respuesta – a la misma consulta – consistente de 16 documentos, *A'*:

$A' = \{\mathbf{d_{79}}, d_{10}, \mathbf{d_{90}}, d_{13}, d_{20}, \mathbf{d_{45}}, d_{60}, d_{30}, \mathbf{d_{77}}, d_{21}, \mathbf{d_{88}}, d_{100}, d_1, \mathbf{d_{91}}, d_{29}, d_{10}\}$

Entonces, resulta:

P = 6 / 16 = 0.38
E = 6 / 10 = 0.60

Y luego, se solicita una nueva respuesta, pero ahora el tamaño de la lista consiste de 20 documentos, *A''*:

$A'' = \{\mathbf{d_{79}}, d_{10}, \mathbf{d_{90}}, d_{13}, d_{20}, \mathbf{d_{45}}, d_{60}, d_{30}, \mathbf{d_{77}}, d_{21}, \mathbf{d_{88}}, d_{100}, d_1, \mathbf{d_{91}}, d_{29}, d_{11}, \mathbf{d_{81}}, \mathbf{d_2}, \mathbf{d_{70}}, \mathbf{d_{82}}\}$

Los nuevos valores para P y E son:
P = 10 / 20 = 0.50
E = 10 / 10 = 1.00

Como se vio anteriormente, es posible evaluar la performance de un sistema bajo distintas situaciones. En nuestro caso particular hemos analizado la efectividad de un mismo sistema para una misma consulta sobre una misma colección para tres tamaños del conjunto de respuestas (12, 16 y 20). Sin embargo, resulta necesario un análisis más detallado calculando las medidas P y E para cada posición d_j. Este brinda una mejor perspectiva del comportamiento del sistema. Veamos para la salida del ejemplo anterior:

j	*A''*	w	E	P
1	**d_{79}**	1	0.10 (1/10)	1.00 (1/1)
2	d_{10}			0.50 (1/2)
3	**d_{90}**	2	0.20 (2/10)	0.66 (2/3)
4	d_{13}			0.50 (2/4)
5	d_{20}			0.40 (2/5)
6	**d_{45}**	3	0.30 (3/10)	0.50 (3/6)
7	d_{60}			0.43 (3/7)
8	d_{30}			0.38 (3/8)
9	**d_{77}**	4	0.40 (4/10)	0.44 (4/9)
10	d_{21}			0.40 (4/10)
11	**d_{88}**	5	0.50 (5/10)	0.45 (5/11)
12	d_{100}			0.42 (5/12)
13	d_{1}			0.38 (5/13)
14	**d_{91}**	6	0.60 (6/10)	0.43 (6/14)
15	d_{29}			0.40 (6/15)
16	d_{11}			0.38 (6/16)
17	**d_{81}**	7	0.70 (7/10)	0.41 (7/17)
18	**d_{2}**	8	0.80 (8/10)	0.44 (8/18)
19	**d_{70}**	9	0.90 (9/10)	0.47 (9/19)
20	**d_{82}**	10	1.00 (10/10)	0.50 (10/20)

En esta tabla se puede apreciar como varia la Precisión a medida que se recuperan más documentos relevantes, es decir, cuando se avanza en ranking de la lista de respuesta y aumenta la exhaustividad. En el gráfico 3 se muestra la evolución de ambas medidas conforme se avanza en la lista rankeada de documentos recuperados. El eje x representa el número de documentos evaluados y las curvas muestran el comportamiento de las medidas.

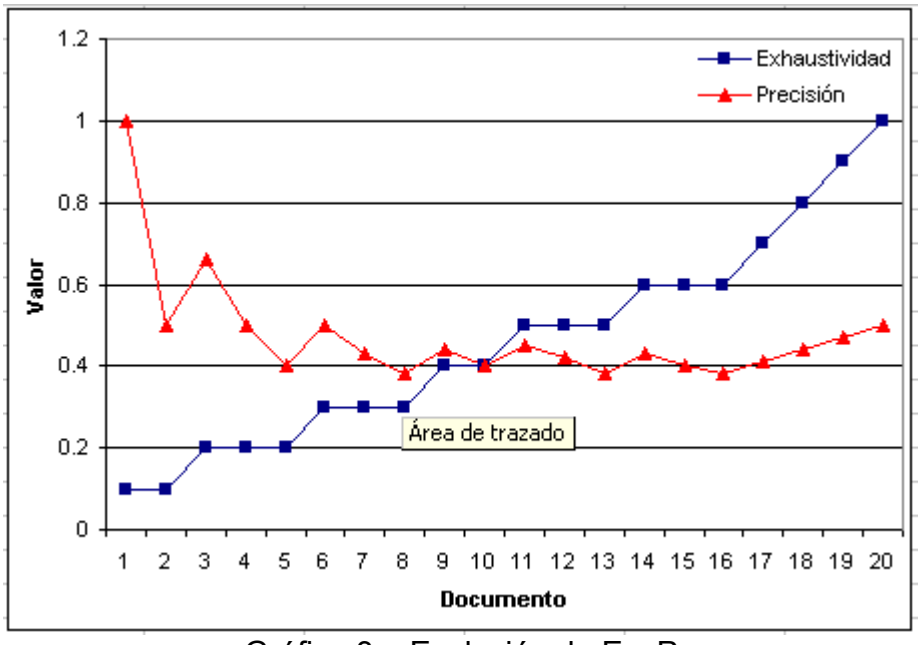


Gráfico 3 – Evolución de E y P

Para esta evaluación también resulta de utilidad una gráfica donde se relacionen ambas medidas. Generalmente, para el eje X se toman 11 niveles standard de Exhaustividad (0.0, 0.1, 0.2,, 0.9, 1.0). Los niveles de exhaustividad se utilizan para mostrar el comportamiento de un sistema de recuperación contrastándolos con la precisión. En el gráfico 4 se muestra esta relación para la salida del sistema del ejemplo anterior.

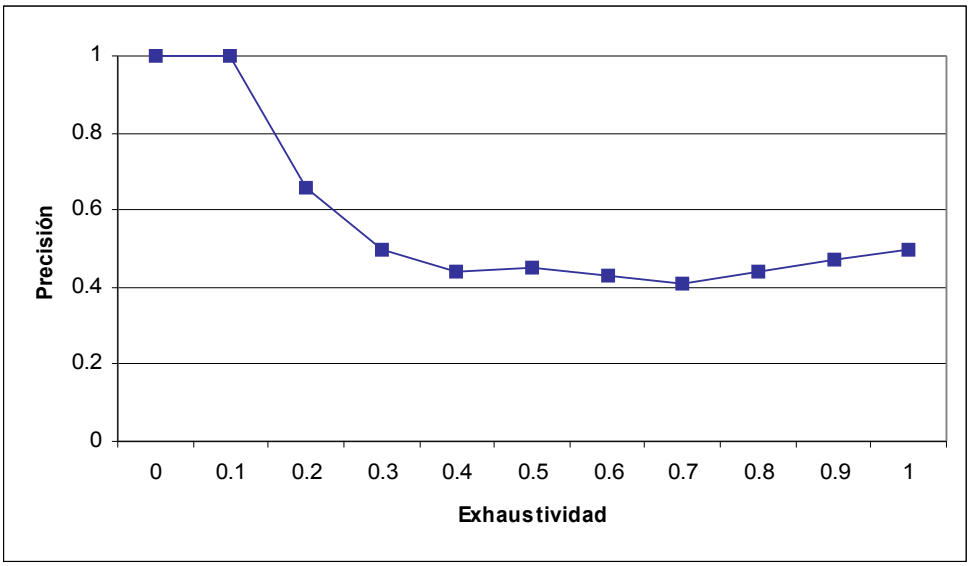


Gráfico 4 – Curva de Exhaustividad/Precisión

Como se puede apreciar, los valores de Precisión y Exhaustividad son relativos al tamaño de la respuesta que se está evaluando y permiten – solo a través del análisis detallado – evaluar minuciosamente la performance del sistema.

Ahora bien, supóngase que para el mismo corpus y la misma consulta otro sistema entrega la siguiente respuesta consistente de 20 documentos, *B''*:

B'' = {**d_{79}**, d_{10}, **d_{90}**, **d_{81}**, **d_{2}**, **d_{70}**, **d_{82}**, d_{13}, d_{20}, **d_{45}**, d_{60}, d_{30}, **d_{77}**, **d_{91}**, d_{21}, **d_{88}**, d_{100}, d_{1}, d_{29}, d_{11}}

Los valores para P y E son:

P = 10 / 20 = 0.50
E = 10 / 10 = 1.00

Por lo que presupone que ambos sistemas tienen una performance equivalente. Sin embargo, si lo analizamos para cada posición de d_j tendremos:

j	*A''*	W	E	P
1	**d_{79}**	1	0.10 (1/10)	1.00 (1/1)
2	**d_{90}**	2	0.20 (2/10)	1.00 (2/2)
3	d_{10}			0.66 (2/3)
4	**d_{81}**	3	0.30 (3/10)	0.75 (3/4)
5	**d_{2}**	4	0.40 (4/10)	0.80 (4/5)
6	**d_{70}**	5	0.50 (5/10)	0.83 (5/6)
7	**d_{82}**	6	0.60 (6/10)	0.86 (6/7)

8	d_{13}			0.75 (6/8)
9	d_{20}			0.66 (6/9)
10	**d_{45}**	7	0.70 (7/10)	0.70 (7/10)
11	d_{60}			0.63 (7/11)
12	d_{30}			0.58 (7/12)
13	**d_{77}**	8	0.80 (8/10)	0.62 (8/13)
14	**d_{91}**	9	0.90 (9/10)	0.64 (9/14)
15	d_{21}			0.60 (9/15)
16	**d_{88}**	10	1.00 (10/10)	0.63 (10/16)
17	d_{100}			0.59 (10/17)
18	d_{1}			0.55 (10/18)
19	d_{29}			0.52 (10/19)
20	d_{11}			0.50 (10/20)

En el gráfico 5 se muestra la curva de Exhaustividad/Precisión para la salida del nuevo sistema (denominado Sistema 2) comparado con el anterior (Sistema 1). Aquí se puede apreciar que este nuevo sistema siempre alcanza una mayor precisión al recuperar los documentos entre los primeros de la lista de respuesta. Esto se debe a que la exhaustividad máxima se alcanza antes de tener que revisar toda la respuesta. Ahora bien, ¿Es mejor el segundo sistema que el primero? En términos de Exhaustividad y Precisión se puede afirmar que sí a partir de este análisis.

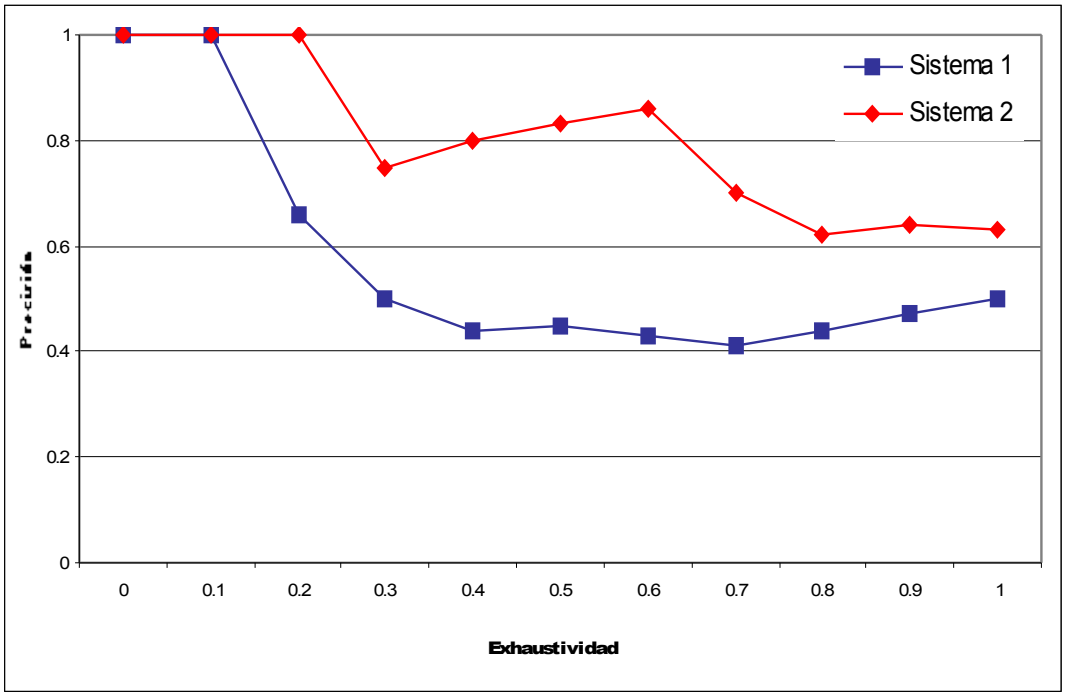


Gráfico 5 – Curva de Exhaustividad/Precisión comparando los dos sistemas

Ahora, supóngase que para una nueva consulta el conjunto de los documentos relevantes es el siguiente:

$R = \{d_2, d_{45}, d_{70}, d_{77}\}$

Si se evalúan los niveles de exhaustividad nos encontramos con 0.25, 0.50, 0.75 y 1.00. Si se desean comparar dos o más consultas resulta necesario normalizar los niveles

de exhaustividad a los standard utilizando interpolación. La precisión interpolada al nivel standard de exhaustividad *j* se define como la máxima precisión obtenida en algún nivel de exhaustividad entre *j* y *(j + 1)*:

$$P(r) = \max_{r_j <= r <= r_{j+1}} P(r) \qquad j \in \{1, 2, 3, \ldots, 9, 10\}$$

Si por una consulta realizada al sistema se tiene la siguiente respuesta:

A' = {**d_{77}**, d_{10}, **d_{70}**, d_{13}, d_{20}, **d_{45}**, **d_2**}

El análisis detallado resulta:

j	*A''*	W	E	P
1	**d_{77}**	1	0.25 (1/4)	1.00 (1/1)
2	**d_{10}**	2	0.50 (2/4)	1.00 (2/2)
3	d_{70}			0.66 (2/3)
4	d_{13}			0.50 (2/4)
5	d_{20}			0.40 (2/5)
6	**d_{45}**	3	0.75 (3/4)	0.50 (3/6)
7	**d_2**	4	1.00 (4/4)	0.57 (4/7)

Los 11 valores interpolados son:

E	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
P	1	1	1	1	1	1	0.66	0.66	0.57	0.57	0.57

Y la gráfica correspondiente:

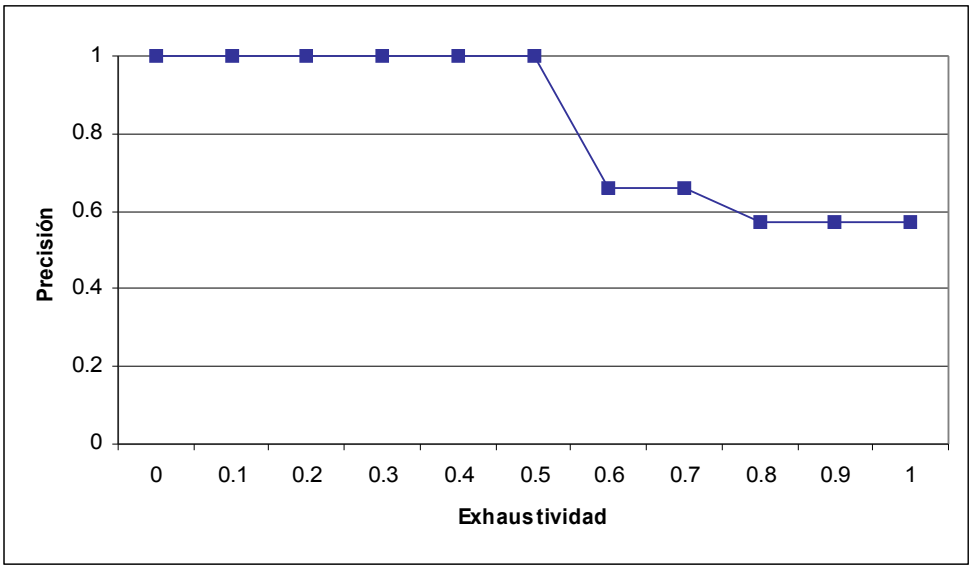


Gráfico 6 – Curva de Exhaustividad/Precisión interpolada

Si bien aquí se presentaron los resultados para una sola consulta, en la evaluación de un sistema real se deben ejecutar decenas de consultas y promediar los resultados

antes de comparar diferentes sistemas. Para un conjunto Q, de tamaño |Q|, la precisión promedio al nivel r es:

$$\overline{Pr} = \sum_{q \in Q} \frac{P_i(r)}{|Q|}$$

donde $P_i(r)$ es la precisión en el nivel de exhaustividad i-ésimo.

Normalmente, para poder comparar dos o más sistemas se deben ejecutar varias corridas de cada uno, utilizando el conjunto de consultas Q y – finalmente – obtener la precisión promedio.

Aunque la Exhaustividad y la Precisión son ampliamente utilizadas como base para la evaluación de los SRI, Baeza-Yates [2] señala algunas cuestiones referidas a éstas:

1) Para poder determinar la Exhaustividad máxima para una consulta se requiere conocer completamente la colección, al detalle de discernir los documentos relevantes de los que no lo son. Por otro lado, P se puede calcular de manera exacta mientras que E no siempre.

2) Estas medidas capturan aspectos diferentes del conjunto de respuesta y – en algunos casos – resulta más útil una medida única. En esta cuestión Korfhage [Korfhage, 1999] señala que E y P se encuentran relacionadas de tal manera que si se las analiza por separado muestran una vista incompleta de la efectividad del sistema evaluado.

3) Estas medidas requieren del procesamiento por lotes de un conjunto de consultas, por lo que no resultan útiles en sistemas interactivos.

Complementariamente a lo expresado en el punto 2, Martínez-Méndez [36] plantea que muchos usuarios consideran más importante la Precisión ya que – mientras encuentren información relevante – no se preocuparán tanto por los documentos que no se recuperan. Según Cleverdon, la Precisión resulta interesante al usuario, no así la Exhaustividad ya que se valoran más las salidas sin ruido. No obstante, hay situaciones donde un usuario puede estar interesado en valores altos de Exhaustividad. Suponga que un estudiante se encuentra realizando una tesis sobre un tema cualquiera. En la etapa de relevamiento de trabajos relacionados, al interactuar con un SRI, seguramente estará interesado en que el resultado de su búsqueda sean todos los documentos existentes – aunque se conforme con una alta proporción – sobre su tema de trabajo.

Un ejemplo opuesto al presentado en el párrafo anterior, en el cual se desee obtener alta Precisión, es el caso de un usuario que utiliza un buscador de Internet para hallar el significado de un término, donde espera que su necesidad de información se satisfaga en el menor tiempo posible[9].

A continuación, se presentan otras medidas que complementan las enunciadas. Algunas son definiciones nuevas y otras combinaciones de E y P que brindan un valor único de la performance de un sistema.

[9] Vea la opción "Voy a tener suerte" del buscador Google (http://www.google.com).

2.2 – Medidas complementarias

2.2.1 – Precisión-R

Una medida complementaria es la **Precisión-R**, la cual corresponde al valor de la Precisión en la posición R-ésima, donde R es la cantidad de documentos relevantes para una consulta.

Suponga que un sistema retorna la siguiente salida para una consulta q cualquiera (los documentos marcados son los relevantes recuperados), donde la cantidad de documentos relevantes para q es 8.

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Doc	d_{98}	d_{24}	d_{22}	d_{2}	d_{23}	d_{101}	d_{43}	d_{65}	d_{51}	d_{5}
Rel	X	x			x		x			x

↑
R=8

Ahora, se calcula la Precisión en la posición R (j = 8) para obtener esta medida, resultando:

$$\text{Precisión - R} = \frac{4}{8} = 0.5$$

Un valor de **Precisión-R** de 1.0 corresponde a una recuperación con Precisión y Exhaustividad perfecta la cual – como hemos mencionado – no es una situación normalmente alcanzable.

Esta medida es útil para analizar un sistema según las respuestas entregadas consulta por consulta. Para ello, se ejecutan un conjunto de tareas de recuperación para |Q| *queries* y se calcula la Precisión-R para cada una. Luego, se promedian para obtener una medida general del sistema. Otra posibilidad es utilizar esta medida para comparar dos sistemas a través de histogramas producidos con el conjunto Q de consultas. Sean $PR_A(i)$ la Precisión-R de un sistema A y $PR_B(i)$ la Precisión-R de un sistema B, para una consulta i, se define la diferencia [Baeza-Yates, 1999]:

$$PR_{A/B}(i) = PR_A(i) - PR_B(i)$$

Luego, si:

$PR_A(i) > 0$, el sistema A supera al B
$PR_A(i) < 0$, el sistema B supera al A
$PR_A(i) = 0$, ambos sistemas se comportan de manera equivalente

En la siguiente tabla se muestra un ejemplo sobre 10 consultas realizadas a dos sistemas (A y B). El gráfico 7 corresponde al histograma de precisión ejemplo para los dos sistemas. Aquí se aprecia claramente la cantidad de *queries* en que uno superó al otro.

Q	$PR_A(i)$	$PR_B(i)$	$PR_A(i) - PR_B(i)$
1	1.00	0.50	0.50
2	0.90	0.50	0.40
3	0.20	0.90	-0.70
4	0.90	0.10	0.80
5	0.70	0.75	-0.05
6	0.50	1.00	-0.50
7	0.60	0.50	0.10
8	0.60	0.40	0.20
9	0.60	0.80	-0.20
10	0.90	0.00	0.90
P(PR)	**0.69**	**0.55**	

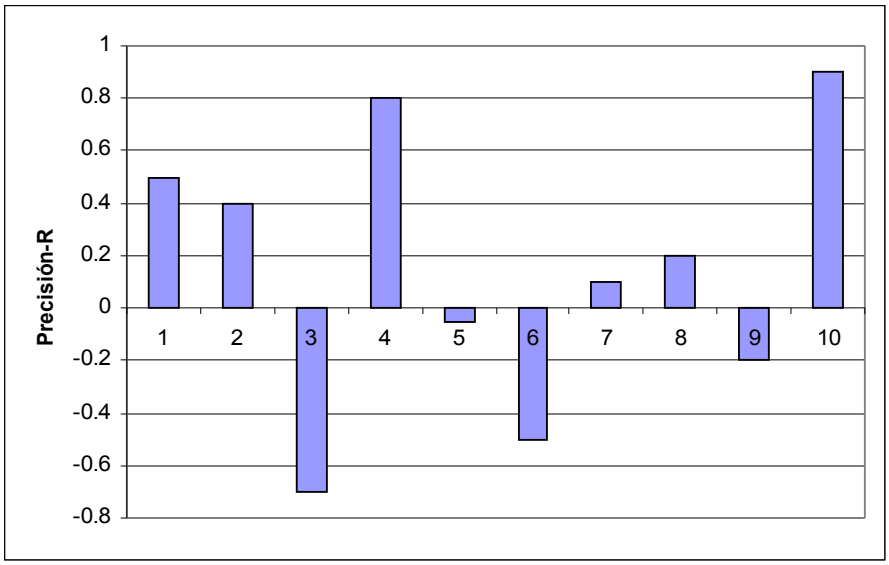


Gráfico 7 – Histograma de Precisión-R

Si se analizan por consulta se puede observar que el sistema A superó en 6 *queries* al sistema B, mientras este último lo realizó en 4 oportunidades con el sistema A. Además, si se promedian las diferencias $PR_A(i) - PR_B(i)$ se tendrá un valor positivo en función de lo explicado anteriormente. Esta evaluación indica que el sistema A tiene una mejor performance en la recuperación que el sistema B.

2.2.2 – Score-F o Media Armónica

Esta medida combina la Precisión y la Exhaustividad en un único valor, también entre 0 y 1. Lo interesante de esta métrica es que un máximo valor de F corresponde al mejor compromiso entre P y E y su valor solamente será alto cuando ambas componentes tengan valores altos. Si F = 0 no se han recuperado documentos relevantes, mientras que si F = 1 se han recuperado todos los documentos relevantes (y solo estos). Entonces, la media armónica se define como:

$$F(j) = \frac{2}{\frac{1}{e(j)} + \frac{1}{P(j)}}$$

Donde, e(j) corresponde a la Exhaustividad en el punto j y P(j) es la Precisión del j-ésimo documento. Volviendo al ejemplo anterior donde se tenía la siguiente salida:

j	*A''*	W	E	P
1	$\mathbf{d_{77}}$	1	0.25 (1/4)	1.00 (1/1)
2	$\mathbf{d_{10}}$	2	0.50 (2/4)	1.00 (2/2)
3	d_{70}			0.66 (2/3)
4	d_{13}			0.50 (2/4)
5	d_{20}			0.40 (2/5)
6	$\mathbf{d_{45}}$	3	0.75 (3/4)	0.50 (3/6)
7	$\mathbf{d_{2}}$	4	1.00 (4/4)	0.57 (4/7)

Podemos calcular F(j) tal como:

$$F(1) = \frac{2}{\frac{1}{0.25} + \frac{1}{1}} = \frac{2}{4+1} = \frac{2}{5} = 0.40 \qquad F(2) = \frac{2}{\frac{1}{0.50} + \frac{1}{1}} = \frac{2}{2+1} = \frac{2}{3} = 0.66$$

y así sucesivamente. Si lo calculamos para todos los valores de j obtenemos:

j	1	2	3	4	5	6	7
F(j)	0.400	0.667	0.569	0.500	0.444	0.600	0.726

El mejor compromiso entre E y P lo tenemos al recuperar el séptimo documento, por lo tanto es una indicación acerca de cuántas respuestas se pueden solicitar a un SRI.

2.2.3 – Medida-E

La Medida-E también combina E y P pero con la posibilidad de ponderar la importancia relativa de uno u otro y se calcula para un tamaño determinado de respuesta. Se define como:

$$E(j) = \frac{1+b}{\frac{b^2}{P(j)} + \frac{1}{e(j)}}$$

Donde, *e(j)* corresponde a la Exhaustividad en el punto *j* y *P(j)* es la Precisión del j-ésimo documento. Luego, el parámetro *b* es el que permite definir si se pondera más la exhaustividad o la precisión:

Si $b = 1$, ambas tienen igual importancia.
Si $b < 1$ se pondera más la Exhaustividad.
Si $b > 1$ se pondera más la Precisión.

Además de la exhaustividad y la precisión es posible calcular otras medidas [41] [32] a partir de relaciones surgidas de la tabla de contingencia presentada anteriormente.

2.2.4 – Tasa de Fallo (*Fallout*)

La ***tasa de fallo*** determina el porcentaje de documentos recuperados no relevantes sobre el total de documentos no relevantes del corpus.

$$F = \frac{no\_relevantes\_recuperados}{total\_no\_relevantes} \qquad F = \frac{y}{w + y}$$

Volviendo al ejemplo anterior y suponiendo que el corpus total tiene 10 documentos y se recuperaron 7, de los cuales 4 son relevantes y 3 no relevantes, resulta:

$$F = \frac{3}{6} = 0.5$$

Nótese que se recuperaron todos los documentos relevantes ya que se alcanzó un valor de E = 1. Por lo tanto, el sistema ha recuperado un 50% de los documentos no relevantes.

2.2.5 – Ruido (Noise)

El **ruido** determina la proporción de documentos irrelevantes hallados en el juego de documentos recuperados.

$$R = \frac{no\_relevantes\_recuperados}{total\_de\_recuperados} \qquad R = \frac{y}{w + y}$$

Se lo puede interpretar como la medida opuesta a la Precisión, por lo que su valor será complementario de ésta. Para el mismo ejemplo, donde P = 0.57 (ya que se recuperaron 4 documentos relevantes sobre un total de 7), el ruido es:

$$R = \frac{3}{7} = 0.43$$

2.2.6 – Generalidad (Generality)

La **generalidad** corresponde a la proporción de documentos relevantes existentes en el corpus sobre todos los documentos del corpus. Una colección con un valor de generalidad alto, para una consulta, tiene mayoría de documentos relevantes que irrelevantes.

$$G = \frac{total\_de\_relevantes}{total\_de\_documentos} \qquad G = \frac{w + x}{w + x + y + z}$$

Esta medida es más representativa del corpus que del sistema de recuperación de información. Si se promedian varias consultas, brinda una indicación acerca de la relevancia general del corpus a un conjunto de consultas.

2.3 – Colecciones de Prueba

Hasta aquí, hemos planteado diferentes medidas para evaluar la *performance de la recuperación* de un SRI, tanto para sistemas de producción existentes o sistemas experimentales con nuevas estrategias. Para la evaluación se requiere contar con colecciones conocidas sobre las cuales se puedan determinar consultas y la relevancia de los documentos respecto de éstas, para luego calcular las métricas. Estas colecciones se fueron desarrollando con el tiempo y evolucionaron en tamaño y calidad. De manera genérica, nos referimos a estas colecciones como *Colecciones de Prueba* y – básicamente – están formadas por tres componentes, a saber:

1) Un conjunto de documentos que constituyen el corpus.
2) Un conjunto de necesidades de información (NI).
3) Juicios de relevancia que relacionan las NI con los documentos del corpus que son relevantes a éstas.

Una colección de prueba es una herramienta experimental indispensable para los investigadores en RI ya que permite comprender la naturaleza de los resultados, compararlos con otros y reproducir pruebas en iguales condiciones. Los primeros esfuerzos en su creación se deben a Cleverdon, en los denominados *Experimentos Cranfield* en el área aeronáutica entre 1957 y 1968. Si bien estas primeras colecciones contenían unos cientos de documentos, marcaron una línea de trabajo, la cual – en la actualidad – se considera una tradición en la evaluación de los SRI (La Tradición Cranfield).

Posteriormente, los experimentos de Salton en la Universidad de Cornell con el sistema SMART[10] entre 1964 y 1988 también contemplaron la creación de colecciones de prueba, en este caso superando el millar de documentos. En la tabla 1 se presentan algunas colecciones standard (clásicas en RI) con sus características principales. Cabe destacar que los documentos de estas colecciones son de poca extensión (la colección TIME es de noticias, mientras que las demás son resúmenes). Por ejemplo, en la CACM el vocabulario contiene 10446 términos, con un promedio de aproximadamente 40 por documento.

La construcción de colecciones de prueba no es una tarea trivial y plantea algunas cuestiones que pueden aumentar – aún más – la complejidad. La primera de las cuestiones que aparece es cómo identificar los documentos relevantes. En general, la decisión de la relevancia o no de un documento respecto de una consulta es de un asesor humano. Por lo tanto, en colecciones grandes esta tarea puede ser extremadamente costosa. Además, se debe establecer si la relevancia se evalúa de manera dicotómica: a) es relevante b) no es relevante o bien de una manera más fina con una escala, por

[10] ftp://ftp.cs.cornell.edu/pub/smart

ejemplo: a) no relevante, b) poco relevante, c) relevante, d) muy relevante. Finalmente, resulta importante la cantidad de juicios a obtener, es decir, si habrá un solo juicio (solo un asesor) o muchos (más de un asesor). Esta consideración es importante ya que diferentes asesores pueden plantear desacuerdos y se deberá tener un criterio para discernir esta situación.

Nombre Colección	Cantidad de documentos	Cantidad de consultas	Tamaño (Megabytes)
CACM (Ciencias de la Computación)	3204	64	1.5
CISI (Ciencia)	1460	112	1.3
CRAN (Aeronáutica)	1400	225	1.6
MED (Medicina)	1033	30	1.1
TIME (Noticias)	425	83	1.5

Tabla 1 – Colecciones standard

En la figura 1 se presenta un ejemplo de un documento de la colección CACM, identificado como I 63 y en la figura 2, una consulta de la misma colección (I 22).

```
.I 63
.T
Octal Diagrams of Binary Conception and Their Applicability
to Computer Design Logic
.W
This paper dates back the genesis of binary conception
circa 5000 years ago, and octal diagrams about 4800 years
ago, as derived by the Chinese ancients.
 It analyzes the applicability of binary trinities of the
octal diagrams to modern electronic-digital-computer design
logic.
.B
CACM September, 1959
.A
Li, S. T.
.N
CA590906 JB March 22, 1978  4:47 PM
.X
63   5     63
63   5     63
63   5     63
```

Figura 1 – Ejemplo del documento I 63 de la colección CACM

```
.I 22
.W
```

```
 I am interested in hidden-line and hidden-surface
algorithms for
cylinders, toroids, spheres, and cones.  This is a rather
specialized
topic in computer graphics.
.N
 22. Wally Dietrich (hidden line and surf alg)
```

Figura 2 – Ejemplo de la consulta I 22 de la colección CACM

En la década de 1990, en el Instituto Nacional de Estándares y Tecnología (*NIST, National Institute of Standards and Technology*), se comenzó a promover – liderada por Donna Harman – una conferencia dedicada al tratamiento de distintas áreas de la RI y la construcción de grandes colecciones de prueba (millones de documentos) denominada TREC (*Text REtrieval Conference*)[11]. Desde entonces, la TREC se convirtió en un encuentro anual dedicado a diferentes tareas, denominadas *tracks*, que utilizan distintas colecciones. Inicialmente, comenzaron con dos *tracks*:

- Recuperación "ad-hoc" (*Ad-hoc retrieval*)
- Ruteo (*Routing*)

Pero con el tiempo se anexaron otros tales como:

- Recuperación web (*Web retrieval*)
- Recuperación Interactiva (*Interactive retrieval*)
- Filtrado de texto (*Text filtering*)
- Respuestas a preguntas (*Question-Answering*)
- Recuperación interlenguas (*Cross-Language*)

Las colecciones de la TREC se encuentran formadas por documentos de diferentes fuentes, como por ejemplo: Wall Street Journal (WSJ), Associated Press (AP), Ziff-Davis Computer Archive (ZIFF), Federal Register (FR), US Patentes (PAT), LA Times (LAT), Financial Times (FT). Los documentos se encuentran estructurados con etiquetas SGML para facilitar su tratamiento. En la figura 3 se muestra un ejemplo de un documento:

```
<doc>
<docno> WSJ870324-0001 </DOCNO>
<hl> John Blair Is Near Accord To Sell Unit, Sources
Say</hl>
<dd> 03/24/87</dd>
<do> WALL STREET JOURNAL (J) </so>
<in> REL TENDER OFFERS, MERGERS, ACQUISITIONS (TNM)
MARKETING, ADVERTISING
(MKT) TELECOMMUNICATIONS, BROADCASTING, TELEPHONE,
TELEGRAPH (TEL) </in>
<dateline> NEW YORK </dateline>
<text>
John Blair &amp;amp; Co. is close to an agreement to sell its
TV station advertising representation operation and program
production unit to an investor group led by James  H.
Rosenfield, a former CBS Inc. executive, industry sources
said. Industry sources put the value of the proposed
acquisition at more than $100 million. ...
</text>
```

[11] http://trec.nist.gov/

```
</doc>
```

Figura 3 – Ejemplo de Documento de la TREC

Por otro lado, las colecciones incluyen descripciones de las necesidades de información utilizadas para la evaluación, que en el marco de la TREC reciben el nombre de *topic*. En la figura 4 se muestra un ejemplo de un *topic*. Con estos topics cada participante genera un *query* que es evaluado por su sistema, de acuerdo al tratamiento particular que decida.

```
<top>
<head> Tipster Topic Description
<num> Number: 066
<dom> Domain: Science and Technology
<title> Topic: Natural Language Processing
<desc> Description: Document will identify a type of natural
     language processing technology which is being developed or
     marketed in the U.S.
<narr> Narrative: A relevant document will identify a company or
     institution developing or marketing a natural language
     processing technology, identify the technology, and
     identify one of more features of the company's product.
<con> Concept(s):  1. natural language processing ;2.
     translation, language, dictionary
<fac> Factor(s):
<nat> Nationality: U.S.</nat>
</fac>
<def> Definitions(s):
</top>
```

Figura 3 – Ejemplo de *Topic* de la TREC

La creación de los juicios de relevancia es la tarea más dificultosa debido a que las colecciones modernas poseen millones de documentos y es imposible chequear cada uno con respecto a cada *query* derivado de cada *topic*. En la TREC se definió un mecanismo de combinación (*pooling*) en el cual solo una fracción de la colección se selecciona para evaluar manualmente. La creación de los juicios de relevancia de las colecciones de la TREC requiere de la participación de diferentes grupos de investigación pertenecientes a universidades, laboratorios y empresas. Cada grupo utiliza su propio SRI para consultar las colecciones con los *queries*. El proceso para la creación de los juicios de relevancia es el siguiente:

1) El NIST crea 50 *topics* y los remite a los participantes, quienes crean sus consultas (*queries*) y las procesan contra el conjunto de documentos.

2) Cada participante envía una cantidad de corridas (*runs*) que consisten de – como máximo – los 1000 primeros documentos recuperados para cada tema. Un subconjunto de las corridas de cada participante se define como "*corrida oficial*".

3) El NIST toma los 100 primeros documentos por tema de cada corrida oficial para formar un grupo (*pool*) para cada tema (removiendo los duplicados). Se ha estudiado que con 100 documentos se obtienen resultados confiables, aun cuando algunos documentos relevantes se pierden.

4) En NIST, un asesor humano debidamente entrenado juzga todos los documentos en el *pool* para aquellos temas que él creo. Los resultados son los juicios de relevancia, tradicionalmente denominados *qrels*.

5) Con los qrels, las corridas se evalúan con el software `trec_eval`, el cual reporta ciertas medidas de performance.

Si bien con este método algunos documentos relevantes se pueden perder, como el subconjunto examinado posee una muestra representativa de los documentos relevantes se pueden aproximar los resultados.

Una segunda dificultad al crear los juicios de relevancia es que – a menudo – los asesores humanos no están de acuerdo acerca de la relevancia. Esta situación se ha estudiado por Vorhees [63] y se determinó que tiene poca influencia en la efectividad relativa de los sistemas. Para ello, utilizó varios conjuntos independientes de juicios de relevancia y encontró que mas allá del bajo solapamiento y su amplia variación entre temas particulares, el ranking relativo se mantuvo sin cambios para los diferentes conjuntos.

La TREC fue el primer esfuerzo en crear grandes colecciones de prueba, las cuales proveen resultados más confiables. Debido a que resulta imposible la creación de los *qrels* de manera manual, el método descripto anteriormente resulta adecuado. Sin embargo, este tema continua en discusión y algunos investigadores han propuesto métodos alternativos para crear juicios de relevancia [15] [54].

2.4 – Actividades

1) Se requiere evaluar la performance en la recuperación de un sistema. Para una consulta q_1, dicho sistema entregó la siguiente salida.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	N	N	R	R	N	N	N	N	R	N	N	N	R	N

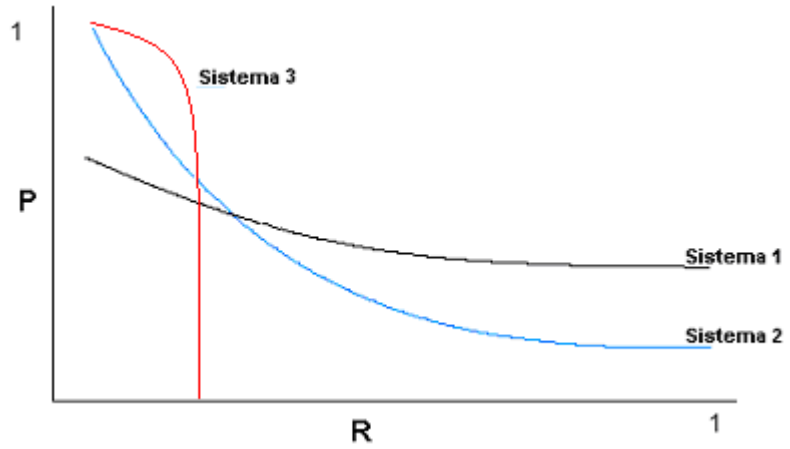
Los documentos identificados como R son los relevantes, mientras que las N's corresponden a documentos no relevantes a q_1. Suponga – además – que existen en el corpus otros 6 documentos relevantes a q_1 que el sistema no recuperó.

A partir de esta salida calcule las siguientes medidas:

a) Recall y Precision para cada posición j
b) Recall y Precision promedio
c) Precisión al 50% de Recall
d) Precisión interpolada al 50% de Recall
e) Precisión-R

Finalmente, realice las gráficas interpolada y sin interpolar. Luego, interprete brevemente los resultados y brinde una explicación.

2) A partir de la siguiente gráfica de R y P brinde un breve explicación del comportamiento de cada sistema. ¿Cuál es mejor? Proponga y justifique – para cada caso – aplicaciones ejemplo donde sería adecuado el comportamiento mostrado del sistema.



3) Utilizando la colección de prueba CISI (http://www.ri.unlu.edu.ar/recursos/) y el software de RI mg (http://www.cs.mu.oz.au/mg/) se debe realizar la evaluación del sistema. Para ello, es necesario construir un índice con los documentos de la colección y luego ejecutar las consultas. Los resultados deben ser comparados contra los juicios de relevancia de la colección utilizando el software trec_eval (http://trec.nist.gov/trec_eval/). Realizar el análisis y escribir un reporte indicando los resultados obtenidos, junto con la gráfica de R–P en los 11 puntos standard.

4) Dadas las salidas de los siguientes sistemas evalúe la performance de cada uno utilizando las medidas F y E. Suponga que existen 10 documentos relevantes para el query.

S1	S2	S3	S4	S5	S6
R	N	R	R	N	N
R	R	N	R	N	R
N	N	R	R	N	N
N	R	N	N	N	N
N	N	N	N	R	R
R	R	N	N	R	R
N	R	R	N	R	N
N	N	R	N	R	R
N	N	N	N	R	N
R	N	R	N	R	N

¿Cómo se modifica su evaluación si pondera en E la Precisión al doble de la Exhaustividad? ¿Y si es al triple? Realice – en cada caso – la gráfica correspondientes.

5) Se presentan a continuación dos narrativas que corresponden a necesidades de información de un usuario cualquiera. A partir de éstas, debe crear – por cada una – dos consultas (queries) diferentes. Para crear los queries puede incluir términos que no se encuentren en las descripciones y – eventaulamente – escribir las consultas en inglés.

N1: *Se necesitan datos acerca de casos de personas atacadas por diferentes especies de tiburones (específicamente).*

N2: *¿Qué atletas obtuvieron records de velocidad en natación en cualquier estilo? Se necesitan documentos que mencionen al atleta, el estilo y su record.*

Luego ejecute las consultas en el metabuscador Dogpile (http://www.dogpile.com) y analice los 20 primeros resultados de cada corrida, juzgando – a su criterio – si el documento es relevante a la necesidad de información o no. Arme una tabla con las siguientes columnas: Q#, D#, JR

Donde Q# es el número de consulta (1..4), D# es el número de documento (1..20) y JR es su juicio de relevancia (R – Relevante, N – No relevante).

Finalmente, ejecute nuevamente las consultas en los buscadores Google y Yahoo y evalue las 50 primeras respuestas contra sus juicios de relevancia. Calcule la Precisión cada 10 documentos y en los 11 niveles standard de Recall. ¿Qué sistema es mejor? ¿Puede justificar su respuesta? ¿Qué debilidades presenta su evaluación?

Capítulo 3

Preprocesamiento

Para poder implementar mecanismos de recuperación sobre una colección de documentos de texto es necesario obtener una representación de los mismos. Dicha representación responde a la implementación de un conjunto de criterios mediante los cuales se obtienen los términos y las relaciones entre éstos. Toda implementación de un SRI comienza con esta tarea de procesamiento del corpus. Esto se debe a que no todos los términos que componen un documento son igualmente representativos de su contenido. Cuestiones como su posición, la cantidad de ocurrencias o su función lingüística – entre otras – definen el grado de importancia de cada unos de los términos. El resultado es una representación de la colección, que es computacionalmente adecuada para los procesos siguientes y que – generalmente – se describe como "indexación de la colección".

En este capítulo se desarrollan – exclusivamente – las técnicas de análisis de texto que componen el proceso de indexación de una colección de documentos, dejando para el próximo capítulo lo relacionado con las estructuras de datos asociadas.

3.1 – El proceso de indexación

La indexación es una operación que tiene por función la identificación de los conceptos que representan el contenido de un documento y la traducción de los mismos a una forma que computacionalmente sea manejable.

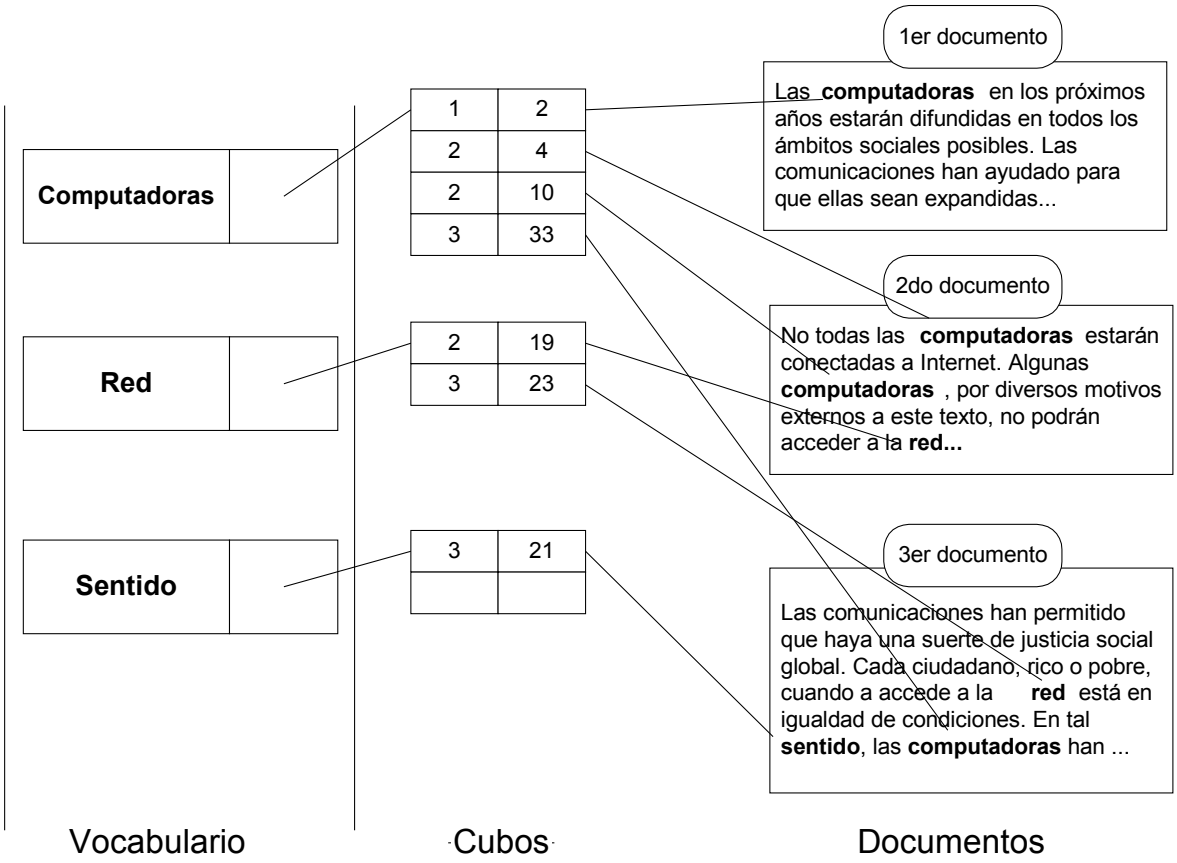
Van Slype [62] plantea que la tarea de indexar consiste de tres etapas, a saber:

a) Familiarización con el contenido del documento.
b) Análisis documental.
c) Selección de los términos más representativos de su contenido.

Sin embargo, Lancaster [32] identifica solo dos etapas:

a) Reconocimiento del documento y extracción de los conceptos contenidos en el mismo.
b) Traducción de tales conceptos en términos de un lenguaje documental.

En el área de RI de manera automática, el concepto de indexación incluye la construcción de estructuras de datos que permitan almacenar tales términos representativos para soportar – posteriormente – la recuperación.



En el gráfico anterior se observa cómo luego del análisis de un fragmento de texto se seleccionaron términos representativos y se estructuraron bajo un criterio, manteniendo tanto la relación con el documento original como así también su posición interna.

3.2 – Enfoques de la indexación

El proceso de indexación puede realizarse desde dos enfoques: uno basado en métodos no lingüísticos y otro basado en métodos lingüísticos. En el primer caso, se utilizan técnicas estadísticas para análisis de frecuencias y cálculo de pesos de los términos, análisis de probabilidades para determinación de multipalabras y técnicas de agrupamiento (clustering) destinadas a la detección y extracción de relaciones. En el segundo caso, se utilizan técnicas derivadas del procesamiento del lenguaje natural (PLN), las que pretenden imitar el comportamiento de los indizadores humanos. En este capítulo, es de principal interés el abordaje de los métodos no lingüísticos, aunque se realiza una breve introducción a los métodos lingüísticos.

Existen diversas técnicas que se pueden utilizar basadas en el enfoque lingüístico. Si bien algunas de éstas no se encuentran completamente desarrolladas para la comprensión del lenguaje natural y aún no permiten la construcción de una representación perfecta de los documentos, se utilizan en sistemas de RI [24]. Las técnicas más comunes son:

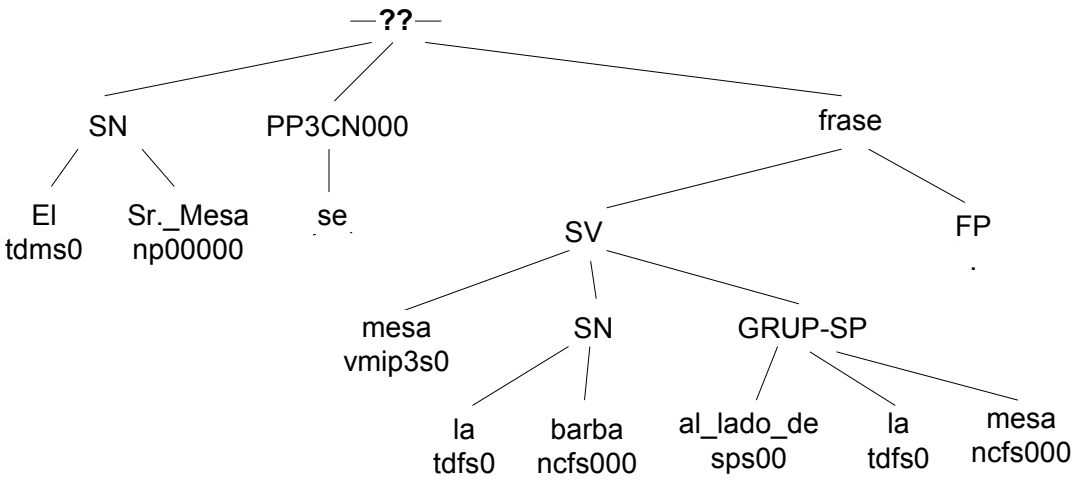
a. ***Procesamiento morfológico-léxico***: Se trata de identificar formas sintagmáticas, siglas y locuciones. Esta técnica tiene como función principal obtener el léxico, el cual es el componente fundamental en los análisis posteriores sintáctico y semántico. El analizador morfológico permite que el análisis estadístico de frecuencias se realice sobre datos normalizados. Nótese que la idea principal del procesamiento morfológico-léxico es convertir un flujo de caracteres a un flujo de palabras, para lo cual deberá tener técnicas para tratar los números, guiones, signos de puntuación, acrónimos, etc.

Una herramienta comúnmente utilizada son los etiquetadores de categorías gramaticales (Part of Speech Tagger) que tienen las funciones de asignar automáticamente la categoría léxica y brinda información sobre las categorías gramaticales. Un ejemplo de salida de procesamiento para la oración "El gato come pescado" es el siguiente:

```
El        el TDMS0
Gato      gato NCMS000
Come      comer VMIP3S0, comer VMMP2S0
Pescado   pescado NCMS000, pescar VMP00SM
```

Una demostración de las capacidades de un etiquetador se encuentra en la página del Grupo de Lingüística Computacional de la Universidad de Barcelona, http://www.ub.es/gilcub/lascosas/eines/esinfo.html

b. ***Procesamiento sintáctico***: Los analizadores sintácticos determinan la construcción de las oraciones localizando la función que cumplen las palabras como sujeto, verbo, complemento. El objetivo principal es describir la estructura de las oraciones que componen los documentos. En el análisis sintáctico se separan las unidades lingüísticas con sentido simple o compuesto y se desambiguar las categorías gramaticales asignadas por el analizador morfológico. Una salida típica de un analizador sintáctico, en la cual se muestran las funciones sintácticas tiene la siguiente forma:



Una demostración de una analizador sintáctico se encuentra en la página de la empresa Connexor, cuya dirección en Internet es http://www.connexor.com/demos/syntax_es.html

c. ***Procesamiento semántico***: El objetivo es obtener el significado de las palabras y – a partir de éstas – de las oraciones que forman. Esto se logra – por ejemplo – mediante el uso de tesauros de términos donde se tienen conceptos y distintos tipos de relaciones éstos. Un ejemplo de herramienta de esta área es WordNet, un sistema de referencia léxica que organiza sustantivos, verbos, adjetivos y adverbios en conjuntos de sinónimos que representan un concepto léxico subyacente. Suponga la búsqueda del termino “baby” en Wordnet. Como respuesta se obtienen los siguientes nodos, denominados “synsets”, donde cada uno hace referencia al término buscado en un contexto particular.

1. baby, babe, infant -- (a very young child (birth to 1 year) who has not yet begun to walk or talk; "isn't she too young to have a baby?")

2. baby, sister -- (sometimes used as a term of address for attractive young women)

3. baby -- (a very young mammal; "baby rabbits")

4. baby -- (the youngest member of a group (not necessarily young); "the baby of the family"; "the baby of the Supreme Court")

5. child, baby -- (an immature childish person; "he remained a child in practical matters as long as he lived"; "stop being a baby!")

6. baby -- (a project of personal concern to someone; "this project is his baby")

De la base de datos Wordnet, existe una versión Europea denominada EuroWordnet que adicionalmente funciona como un diccionario multilingüe, vinculando el euskara, el castellano, el catalán, el inglés y el francés.

Si bien estas técnicas lingüísticas asisten al proceso de indexación, al momento, no han aportado mejoras significativas a los sistemas de recuperación de información respecto de las técnicas no lingüísticas, se espera que conforme evolucione el procesamiento del lenguaje natural, los métodos lingüísticos brinden capacidades que permitan lograr mejores desempeños en el área de RI.

3.3 – Indexación en base a técnicas no lingüisticas

Como se mencionó, un documento indexado es una representación del documento original. En la práctica, consiste en una lista de términos o conceptos normalizados, de alto valor semántico, con información adicional asociada (por ejemplo, su frecuencia de aparición o posición en el texto). Los términos pertenecientes al índice pueden estar en su forma original o lematizados y pueden ser palabras simples, multipalabras, siglas o nombres propios.

En general, la indexación de base no lingüística se fundamenta en el análisis de la frecuencia de los términos y su distribución dentro de los documentos. Este análisis tiene como objeto establecer criterios que permitan determinar si una palabra es un término de indexación válido, fundamentalmente porque permite discriminar el contenido de los documentos y – de alguna manera – aporta información. Para ello, se han estudiado y determinado algunas propiedades estadísticas del texto escrito que determinan cómo se distribuyen las frecuencias de aparición de las diferentes palabras en una colección y cómo crece el tamaño del vocabulario conforme crece tal colección. Existen dos leyes empíricas que describen estas propiedades: la ley de Zipf y la ley de Heaps, que presentamos a continuación.

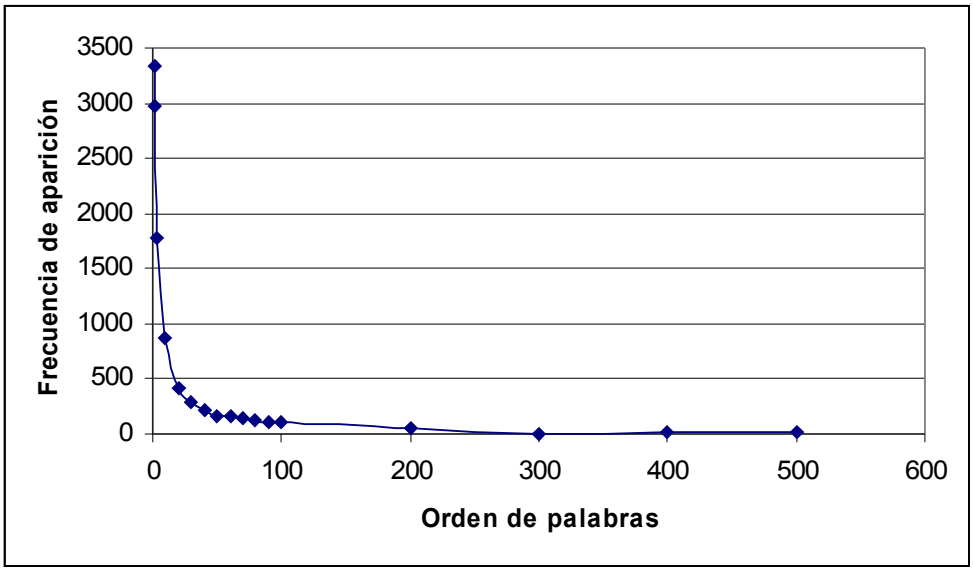
3.3.1 – Ley de Zipf

Además de la eliminación de palabras vacías el vocabulario de una colección puede ser podado utilizando otros criterios. Uno de los más utilizados corresponde a la denominada “Ley de Zipf” [66], quien realizó una serie de estudios empíricos en la década de 1940 que demostraron que la gente al escribir – normalmente – suele preferir palabras más conocidas sobre aquellas menos conocidas. A esto lo denominó se lo conoce como la ley del menor esfuerzo.

Zipf descubrió que si se armaba una lista con las palabras, junto con su cantidad de ocurrencias (en documentos en inglés), y se la ordenaba por frecuencia de mayor a menor, se cumplía que la frecuencia de la i-esima palabra multiplicada por i (el ranking), era igual a una constante C, es decir:

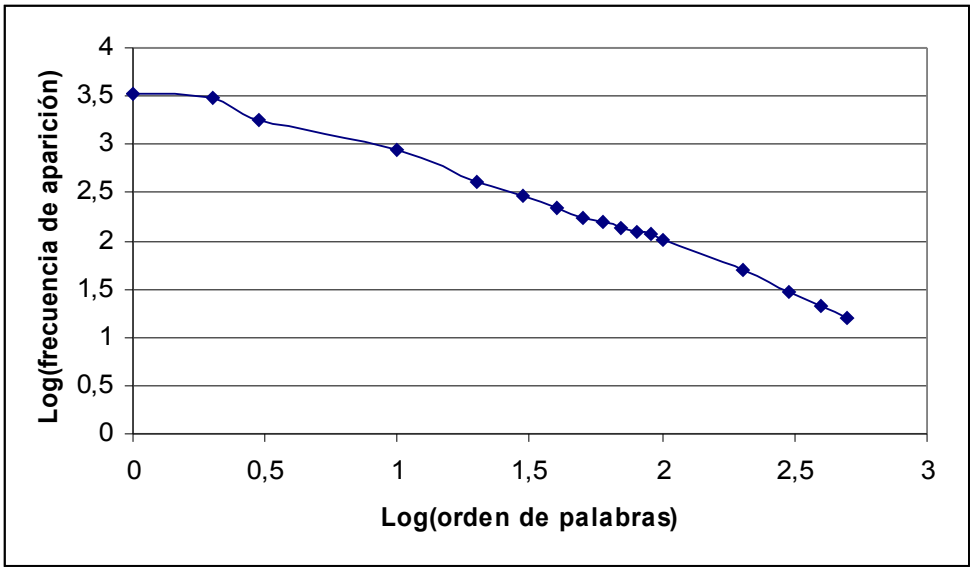
C = ranking del término t * frecuencia termino t

Hoy en día, para documentos recientemente escritos, esta ley se cumple si se eleva i a un exponente mayor que 1 (en textos en inglés es 1.8), lo cual indica una creciente pobreza en la utilización de la lengua.



Gráfica de representación de la frecuencia de términos del libro Tom Sawyer

Al graficar la curva utilizando una escala logarítmica en x e y, se obtiene una recta con pendiente negativa.



Gráfica logarítmica de representación de la frecuencia de términos del libro Tom Sawyer

En la tabla siguiente se muestra el cálculo de la constante C sobre algunos términos que forman el vocabulario del libro Tom Sawyer. Nótese que en los primeros lugares aparecen términos del conjunto de las palabras vacías.

Palabra	**Frecuencia (*f*)**	**Orden (*o*)**	***C* = *f* * *o***
The	3332	1	3332
and	2972	2	5944
a	1775	3	5235
He	877	10	8770
but	410	20	8400
Be	294	30	8820
there	222	40	8880
one	172	50	8600
about	158	60	9480
more	138	70	9660
never	124	80	9920
Oh	116	90	10440
two	104	100	10400

Palabra	**Frecuencia (*f*)**	**orden (*o*)**	***C* = *f* * *o***
turned	51	200	10200
you ’ll	l30	300	9000
name	21	400	8400
comes	16	500	8000
group	13	600	7800
lead	11	700	7700
friends	10	800	8000
begin	9	900	8100
family	8	1000	8000
brushed	4	2000	8000
sins	2	3000	6000
Could	2	4000	8000
Applausive	1	8000	8000

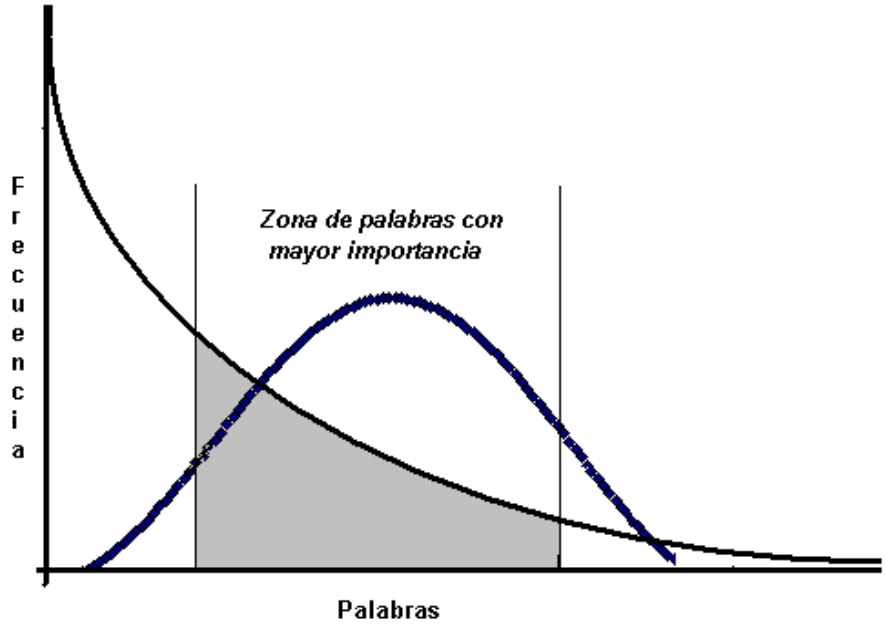
Evaluación Empírica de la Ley de Zipf

Otros trabajos de investigación han demostrado que la ley de Zipf se aplica a otras situaciones relacionadas con la recuperación de información. Para Baeza Yates [http://www.dcc.uchile.cl/~rbaeza/inf/zipf.html] en el espacio web existen fenómenos como los siguientes:

- Tamaños de los archivos que transfiere el protocolo HTTP.

- Número de enlaces que salen de una página.
- Número de enlaces que llegan a una página.
- Fecha de actualización de las páginas, existen más páginas nuevas o modificadas que viejas.
- Frecuencia de acceso a páginas web.

El comportamiento del vocabulario de acuerdo a la ley de Zipf brinda indicios acerca de la utilidad de los términos. En 1958, Luhn [34] sugirió que las palabras que describen de mejor forma el contenido se encuentran en un área comprendida entre las altamente frecuentes y las muy raras (baja frecuencia) y lo ilustró con la siguiente gráfica:



Las divisiones verticales definen una zona de transición entre las palabras de frecuencia muy alta y las de muy baja. Aquí se encuentran los términos con mayor contenido semántico de un documento.

El límite superior corresponde – generalmente – al comienzo de las palabras vacías y – como mencionamos – no se indexan por no tener poder de discriminación. Por otro lado, el limite inferior corresponde al comienzo de las palabras más raras, y no se incluyen en el vocabulario debido a que existe una baja probabilidad que el usuario las use en una consulta. Estas palabras de baja frecuencia son las que denotan la riqueza y el estilo de vocabulario del autor o bien, corresponden a errores de ortografía y para establecer su frecuencia límite se sugiere: a) Eliminar aquellos términos que estén en 3 o menos documentos y b) Eliminar todas las palabras que ocurren una o dos veces [39].

En el anexo 5 se muestra un ejemplo más completo de análisis estadístico de un texto en español.

3.3.2 – Ley de Heaps

De manera similar a la ley de Zipf, existe otra ley empírica que describe el comportamiento de los términos dentro de un texto escrito denominada ley de Heaps. En esta ley, se plantea una relación entre el tamaño del texto (cantidad de palabras) y el crecimiento del vocabulario (cantidad de palabras únicas). En particular, postula que el tamaño del vocabulario (y su crecimiento) es una función del tamaño del texto.

$$V = K \bullet N^{\beta}$$

donde:

N: Es el tamaño del documento (cantidad de palabras)

K: Constante que depende del texto, típicamente entre 10 y 100.

β: También es una constante que depende del texto, donde $0 < \beta < 1$, aunque sus valores típicos se encuentran entre 0.4 y 0.6.

Por ejemplo, para textos en inglés, se toman – aproximadamente – los siguientes parámetros:

$10 \leq K \leq 20$

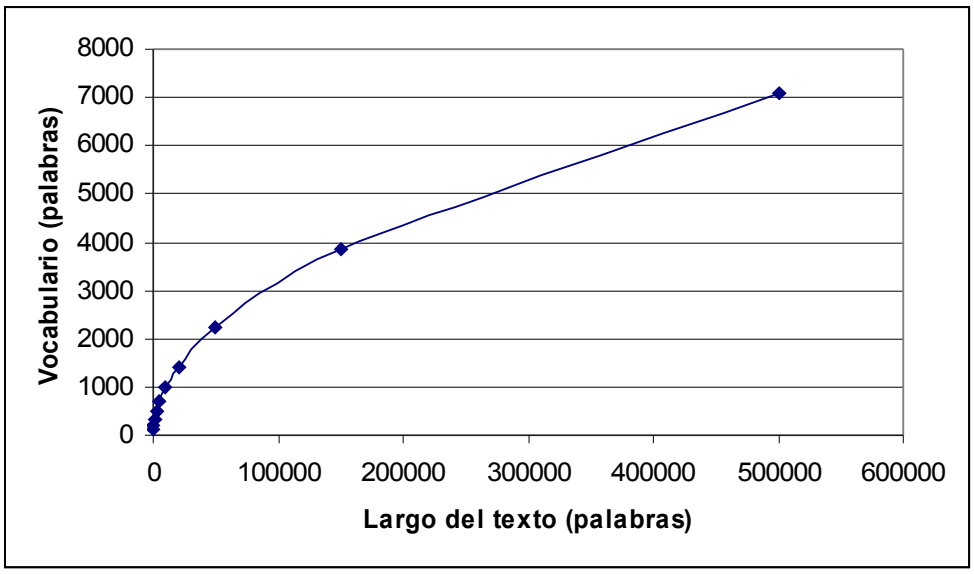
$0.5 \leq \beta \leq 0.6$

Por lo tanto, si $K = 20$ y $\beta = 0.5$, resulta:

N	V
100000	6325
250000	10000
400000	12649
800000	17889
1000000	20000

Nótese que el tamaño del corpus creció 10 veces, mientras que el vocabulario apenas superó las 3 veces su tamaño inicial.

Una grafica de sus valores para diferentes tamaños de documentos, resulta similar a la siguiente:



Los resultados de la ley de Heaps plantean que a medida que se incorporan documentos a una colección, cada vez se descubrirán nuevos términos para el vocabulario.

Su aplicación es directa ya que permite estimar el tamaño del vocabulario con lo cual se puede determinar – por ejemplo – la escalabilidad de las estructuras de datos necesarias para almacenar los índices que soportan el SRI. Esto es altamente útil si se utilizará una tabla de hash en memoria para el índice.

3.4 – El proceso de indexación

La indexación es un proceso de varias etapas que finaliza con la representación del contenido completo de la colección sobre las estructuras de datos definidas a tal fin. Las etapas que conforman este proceso son las siguientes:

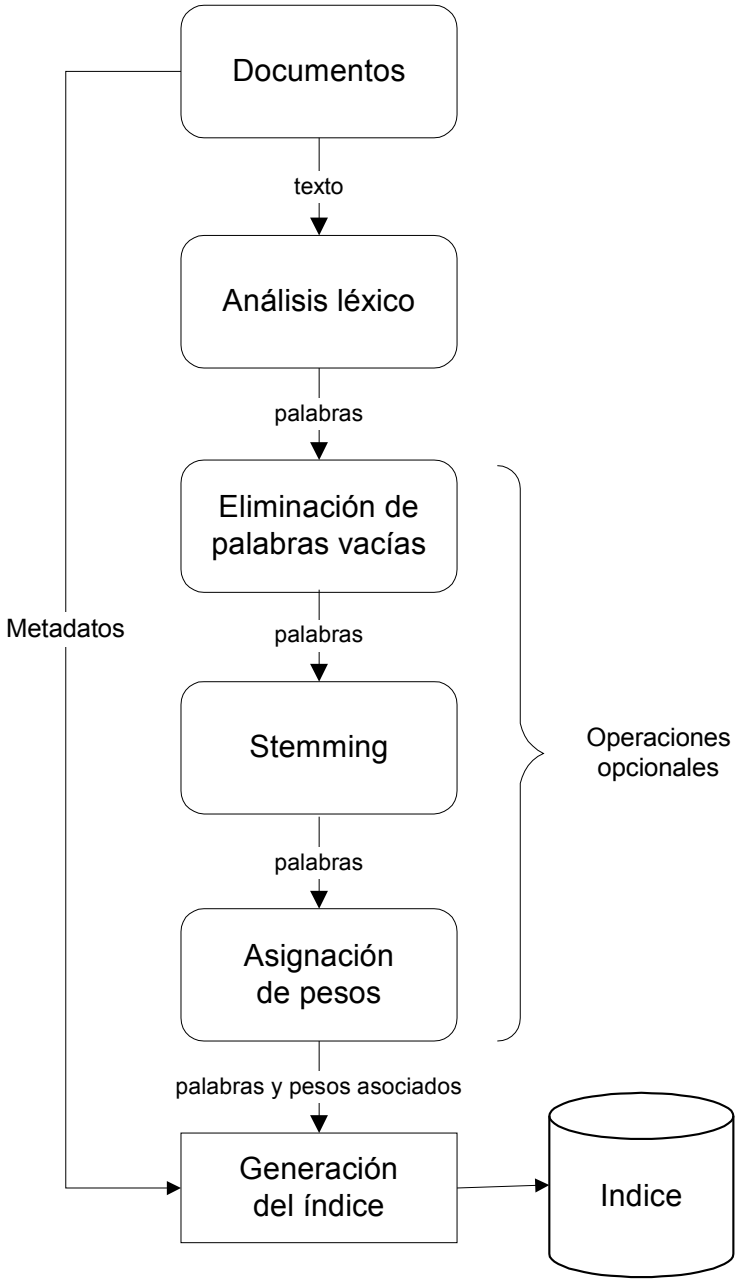


Diagrama de flujo del proceso de indexación

1- Análisis lexicográfico: Se extraen las palabras y se normalizan.

2- Eliminación de palabras vacías o de alta frecuencia en la colección.

3- Stemming: Se reducen palabras morfológicamente parecidas a una forma base, con la finalidad de aumentar la eficiencia de un SRI.

4- Selección de los términos a indexar: Se extraen aquellas palabras simples o compuestas que mejor representan el contenido de los documentos.

5- Asignación de pesos o ponderación de los términos que componen los índices de cada documento. En algunos modelos de RI es fundamental asociar la

importancia de un término t en un documento d a los efectos de mejorar las prestaciones.

Se debe tener en cuenta que esta es una descripción general y que – en algunos casos – se pueden omitir algunas etapas (como la 2 y la 3), mientras que otras varían de acuerdo a las necesidades particulares del SRI en la tarea posterior de recuperación.

3.4.1 – Análisis Lexicográfico

El objetivo del análisis lexicográfico es convertir un flujo de caracteres en palabras o *tokens* (por lo que también recibe el nombre de *tokenization*). Esto implica detectar el comienzo y fin de las palabras bajo diversas circunstancias (al inicio, en el medio o en el fin de una oración). Alternativamente, el analizador debe ser capaz de tratar con símbolos no alfabéticos como dígitos, caracteres especiales que componen las palabras (G**ü**emes – O´Donell) los cuales generalmente se reemplazan por el carácter base relacionado (por ejemplo, **ú**, **ü** se reemplazan por **u**). Además, se debe normalizar y expandir siglas (por ejemplo, **UNLu**, **U.N.L.u** se reemplazan por **Universidad Nacional de Luján**) y tratar guiones como conectores de términos (por ejemplo, MS-DOS, data-base). Luego de realizar este tratamiento todo el texto es – generalmente – transformado a mayúsculas o minúsculas.

Nótese que – si el archivo a procesar posee marcas de formato – también son removidas en esta etapa. Este es el caso de los mensajes de correo electrónico, los archivos latex, las páginas HTML y demás. Sin embargo, se puede dar la situación que algún SRI requiera conservar información sobre algún atributo ligado a oraciones o términos para un uso posterior (por ejemplo, en la etapa de asignación de pesos o de agrupamiento). Algunos atributos que generalmente se conservan son títulos de documentos, autores, títulos y subtítulos de secciones, etc.

En el siguiente ejemplo se puede observar como un analizador léxico trata un flujo de caracteres:

Diagrama ejemplo de flujo de entrada y tranformación realizada

Otra cuestión importante a tener en cuenta es el tratamiento de los números. Su inclusión en el índice depende – generalmente – del tipo de colección y de la importancia que tengan como posibles términos de búsqueda. Por ejemplo, en una colección con normativas, patentes ó histórica (donde hay fechas) resulta importante conservar los términos numéricos. De no ser necesaria esta información, los números son fácilmente eliminados y no aparecen en el índice.

También se debe evaluar si los nombres propios van a formar parte de los términos de indexación. Generalmente, resultan importantes dentro del contenido del documento y – por ende – se sugiere su inclusión. Además, pueden ser ponderados de manera diferente cuando se asignen los pesos. La detección de nombres propios se puede realizar partiendo de la idea que siempre comienzan con una letra en mayúsculas y – luego – chequeándolos con un diccionario de nombres propios para evitar las palabras de comienzo de oraciones.

```
$linea =~ tr/óíáéúüÓÍÁÉÚÜàèìòùÀÈÌÒÙâêîôûÂÊÎÔÛäëïöÄËÏÖåÅãÃõÕ
            /oiaeuuOIAEUUaeiouAEIOUaeiouAEIOUaeioAEIOaAaAoO/;

$linea =~ tr/A-Z/a-z/;
```

Ejemplos de normalización de caracteres en lenguaje Perl

3.4.2 – Eliminación de palabras vacías

Los términos que ocurren en casi todos los documentos de una colección no son buenos para la recuperación de información por su nulo poder para discriminar documentos. Las palabras que aparacen en más del 80% de documentos no deberían seleccionarse como términos de indexación. Este conjunto de términos se lo conoce como palabras vacías o *stopwords*.

Esta categoría de palabras está formada – generalmente – por artículos, preposiciones, conjunciones y forman lo que se conoce como diccionario negativo o anti-diccionario. La lista de palabras vacías depende de cada lenguaje. A continuación se presentan ejemplos para el español y el inglés:

Español
la
las
hasta
sobre
y

Inglés
you
they
many
very
beside

En los anexos 1 y 2 se presentan listas de palabras vacías de la lengua inglesa (extraidas del sistema SMART) y la lengua española.

Una ventaja adicional de la eliminación de éstos términos es la reducción del tamaño de almacenamiento necesario para soportar el corpus. Algunos estudios determinan que se llegan a reducir los archivos de índices en un 40% [21].

La siguiente porción de un programa escritp en lenguaje Perl implementa la función de eliminación de palabras vacías de un archivo de texto:

```
# lectura del archivo que contiene la lista de palabras vacías

open(IN,"vacias.txt");

  while(!eof(IN))
    {
        $linea = <IN>; chop($linea);
        $linea =~ tr/áéíóú/aeiou/;
             $linea =~ tr/A-Z/a-z/;
        $vectorstw[$i++] = $linea;
    }

close(IN);

…
…

# Bloque de instrucciones que elimina palabras vacias contenidas
# en  la variable $archi

foreach $ele (@vectorstw) { $archi =~ s/ $ele / /g; }
```

3.4.3 – Stemming

Es una técnica de reducción (*conflation*) que permite detectar variantes morfológicas de un mismo término, por ejemplo, palabras como cómputo, computadoras, computable, computación son variantes del término computar, y reemplazarlas por el término raíz o lema.

El uso del stemming o lematización posibilita: a) tener índices de menor tamaño y b) una mayor cantidad de respuestas a una consulta dada, debido a que ahora al aplicarse lematización al corpus y a la consulta se recuperan documentos que tengan todas las variantes morfológicas de los términos contenidos en la consulta. Esta última consecuencia puede verse como una desventaja bajo ciertas ocasiones, debido a que aumenta la exhaustividad y disminuye la precisión.

La técnica de stemming permite extraer sufijos y/o prefijos comunes, de tal forma que palabras que literalmente son diferentes, pero tienen una raíz común, pueden ser consideraras como un solo término en base a su raíz. El siguiente ejemplo muestra términos base y algunas de sus posibles variantes:

Término base	**Variantes**
casa	casas, casita, casitas
poder	poderes, poderíos
amable	amables, amabilidad, amabilidades
computar	computadora, computado, computable, computadoras

La reglas que forman los algoritmos clásicos de stemming dependen del idioma de las colecciones de documentos a procesar. El algoritmo clásico es el de Porter [43] cuya versión original está para la lengua inglesa, existen adaptaciones de las reglas para operar en otras lenguas [31] [5].

Básicamente, hay dos problemas que el algoritmo de stemming debe resolver. Uno es el quitar terminaciones de palabras que falsamente parecen ser sufijos. Por ejemplo, mientras que la terminación "ed" sería retirada normalmente de cualquier palabra en otras como "bed" no debería quitarse. Este problema puede ser resuelto fijando un mínimo de letras aceptables para una raíz y con una pequeña lista de palabras que deben ser exentas de la aplicabilidad de esta regla. Si se fijara un mínimo de tres letras para una raíz, la palabra "bed" no sería tocada, pero que pasaría con "breed". Entonces "breed" podría ser candidata para ser excluida de la aplicación de esta regla. Alternativamente se podrían aplicar algunas reglas que determinaran cuándo una raíz debe ser quitada y cuando no. Por ejemplo, incluir una regla que diga que cuándo la palabra termina con "eed", el sufijo "ed" no se quite.

El otro problema que el algoritmo de stemmer debe resolver es el cambio de la raíz en determinadas formas de una palabra. Para clarificar este problema, observe que pasa con el plural de algunas palabras, por ejemplo "Knife" cambiaría en plural por "Knives". Lo mismo ocurre en el castellano con, por ejemplo la palabra "repite" y "repetido". Estos cambios morfológicos son excepciones a las reglas, y en inglés hay relativamente pocos casos donde se dé este problema, en castellano este problema es mayor.

A continuación se muestran las las técnicas básicas de stemming según Frakes [20] :

a. Búsqueda en tabla
b. S-stemer
c. N-gramas
d. Eliminación de sufijos

3.4.3.1 – Búsqueda en tabla

Es un método de rápida computación que requiere de una tabla de doble entrada, donde cada registro corresponda a una variante morfológica asociada a su lema o raíz. Por ejemplo:

Término	Lema
abarca	abarcar
abarcada	abarcar
abarcadas	abarcar
abarcado	abarcar
abarcados	abarcar
abarcan	abarcar
abarcando	abarcar
abarcar	abarcar

Término	Lema
bruta	bruto
bruta	bruto
brutas	bruto
brutas	bruto
bruto	bruto
bruto	bruto
brutos	bruto
brutos	bruto

Una implementación basada en tablas de *hashing* puede brindar un buen desempeño del algoritmo ya que los datos han sido procesados previamente y el la búsqueda se reduce solo a una operación simple de *matching*. El problema de este método se presenta debido a que las lenguas son dinámicas y es necesario realizar actualizaciones periódicas del vocabulario. Por otro lado, en lenguas como el español la

cantidad de variantes de un lema es sustancialmente mayor que en otras como – por ejemplo – el inglés.

3.4.3.2 – S-stemmer

Es un lematizador basado en el número, cuyo única acción es la detectar formas en plural y convertirlas a singular. En la tabla situada a continuación se muestran las reglas para la lengua inglesa. Este proceso se aplica bajo un orden de reglas a palabras de 3 o más caracteres. Se suele utilizar una lista asociada de términos de exepción sobre los cuales no se aplica el proceso.

Orden de aplicación	Reglas	Acción
1	Palabras que finalizan con “ies”, pero no “eies”, o “aies”. - *SI tries -> try* - *SI abilities - > ability* - *SI acuity -> acuities (agudeza)*	Reemplazar “ies” → “y”.
2	Palabras que finalizan con “es”, pero no “aes”, “ees”, “oes” o "ses". - *SI airdromes -> airdrome (aeropuerto)* - *NO appendicitises -> appendicitis*	Reemplazar “es” → “e”
3	Palabras que finalizan en “s”, pero no “us”, or “ss”. - *SI aircrafts -> aircrafts* - *NO airbuses -> airbus* - *NO adventuresses -> adventuress (aventurera)*	Reemplazar “s” → null

Reglas de un algoritmo de S-Stemmer para la lengua inglesa

El siguiente código para procesar con el interprete sed (Stream Editor) – que pertenece al grupo de investigación español Reina[12] – implementa un s-stemmer para la lengua española.

```
#s-stemer para español
#quita plurales
#==================================
# USO:
#       sed -f s_stemmer.sed <documento>
#==================================
# Grupo REINA 2003
# http://reina.usal.es
#-----------------------------------------------
#
#primero, la s final de palabras que
#acaban en vocal
s/\([aeiouAEIOUáéíóúÁÉÍÓÚ]\)[sS]\([^a-zA-ZñÑáéíóúÁÉÍÓÚ]\)/\1\2/g
s/\([aeiouAEIOUáéíóúÁÉÍÓÚ]\)[sS]$/\1/g
#
# la e final de todas las palabras
```

[12] http://reina.usal.es

```
#(no podemos distinguir cuáles eran plurales y cuáles no
s/\([a-zA-ZñÑáéíóúÁÉÍÓÚ]\)[eEéÉ]\([^a-zA-ZñÑáéíóúÁÉÍÓÚ]\)/\1\2/g
s/[eEéÉ]$//
```

Como se ve en el código anterior el algoritmo es incapaz de distinguir sustantivos, adjetivos o verbos irregulares por que trata de la misma manera a todas las palabras.

3.4.3.3 – N-gramas

Antes de presentar el método de n-gramas resulta necesario hacer una breve introducción. Un n-grama es una secuencia de n palabras o caracteres consecutivos que son copiados fuera del texto utilizando una ventana de n palabras o caracteres de longitud, la cual es movida sobre el texto de a una palabra o caracter por vez.

Ejemplo de bigramas (2-gramas) de palabras:

Frase "Mark Knopler sabía que sus sonidos ..."

Bigramas: [Mark Knopler] [Knopler sabía] [sabía que] [que sus] [sus sonidos]

Ejemplo de trigramas (3-gramas) de caracteres:

Frase "Mark Knoffler sabía que sus sonidos ..."

Trigramas: [_Ma] [Mar] [ark] [rk_] [k_K] [_Kn] [Kno] [nop] [opl] [ple] [ler] [er_] [r_s] ...

La caracterización de textos utilizando n-gramas ha sido aplicada al procesamiento de documentos en áreas como detección y corrección de errores ortográficos [40], compresión [65], identificación de lenguajes [55] y recuperación de información [10].

Los n-gramas aplicados a la lematización [1] se basan en la cuenta del número de n-gramas que comparten dos palabras. Esta técnica es relativamente inmune a problemas ocasionados por errores ortográficos. Frakes indicó que este método no es especificamente un lematizador, sino que ayuda a la detección de familias de palabras (clases de equivalencia) a un alto costo computacional.

Los pasos de operación son los siguientes:

- Para cada palabra, generar los bigramas correspondientes.
- Computar la similitud palabra a palabra utilizando el coeficiente de Dice.
- Utilizando una matriz de similitud (distancias) generar agrupamientos (clusters) de palabras cercanas.
- Identificar las raices de cada cluster de palabras con el mismo prefijo.

La semejanza entre dos términos se computa – inicialmente – determinando cuantos bigramas únicos tiene cada término, como se ve a continuación:

statistics	→ st ta at ti is st ti ic cs	9 bigramas
bigramas únicos	→ **at** cs **ic is st ta ti**	7 bigramas únicos
statistical	→ st ta at ti is st ti ic ca al	10 bigramas
bigramas únicos	→ al **at** ca **ic is st ta ti**	8 bigramas únicos

Luego, se debe determinar cuántos términos están en la intersección. En el ejemplo anterior ambos términos comparten 6 digramas **at, ic, is, st, ta, ti.** A continuación, utiilizando el coeficiente de Dice, se computa la semejanza entre ambos términos.

$$S = \frac{2C}{A + B}$$ (2 * 6) / 7 + 8 = 0.80 de similaridad

Donde,

A es el número de bigramas únicos en el primer término.
B es el número de bigramas únicos en el segundo término.
C es el número de bigramas únicos que comparten A y B.

3.4.3.4 – Eliminación de sufijos

Se basa en eliminar prefijos y/o sufijos dejando la raíz del término en base a reglas dependientes del idioma a procesar. El algoritmo de Porter – un clásico en esta área – utiliza una lista de sufijos y un criterio para indicar qué porción de una palabra que se debe eliminar y asi obtener su raíz. Cabe aclarar que no debe confundirse la raíz obtenida con el algoritmo con la raíz lingüística de la palabra. Uno de los problemas que enfrenta este algoritmo es que quita algunas terminaciones que falsamente parecen sufijos. El otro problema que tiene es el cambio de la raíz en determinadas formas de una palabra.

El proceso es iterativo, es decir que – a diferencia de otros algoritmos – quita los sufijos por etapas. En cambio, el algoritmo de Lovins requiere de la definición de todas las posibles combinaciones de sufijos (esta tarea es bastante extensa para el español).

En el algoritmo de Porter la palabra es evaluada por varios juegos de reglas, cada uno con n reglas. Luego, cada regla cuenta con:

1. Un identificador de regla
2. El sufijo a identificar
3. El texto por el cual debe ser reemplazado al encontrar el sufijo
4. El tamaño del sufijo
5. El tamaño del texto de reemplazo
6. El tamaño mínimo que debe tener la raíz resultante luego de aplicar la regla (esto es a los efectos de no procesar palabras demasiado pequeñas)
7. Una función de validación (una función que verifica si se debe aplicar la regla una vez encontrado el sufijo)

Considere, a modo de ejemplo, la siguiente regla perteneciente al algoritmo original de Porter :

106, "ed", LAMBDA, 1, -1, -1, ContainsVowel

Analizándola elemento a elemento se tiene que :

1. 106 es el identificador de la regla

2. "ed" es el sufijo que debe localizar al final de la palabra

3. "LAMBDA" es el texto por el cual se debe reemplazar el sufijo una vez encontrado (en este caso LAMBDA es una constante definida como cadena vacía)

4. "1" es el tamaño del sufijo (Tamaño de "ed" – 1)

5. "-1" es el tamaño del texto que se debe reemplazar por el sufijo, como se trata, en este caso, de una cadena vacía el tamaño sería 0 que restándole 1 sería –1

6. El siguiente "–1" es el tamaño mínimo que debe tener la raíz una vez que le quitemos "ed" (-1 significa que lo quite cuando al menos queden tres caracteres)

7. "ContainsVowels" es la función de validación. En el caso particular de ContainsVowels, verifica que la palabra sin "ed" contenga vocales

Esta regla se aplica a las palabras *lowed* quedando *low*, *shared* quedando *shar*, pero no se aplicaría a la palabra *shed*, ya que si le quita "*ed*" no quedarían vocales y – además – la raíz quedaría con 2 letras y se exige que tenga al menos 3.

Los algoritmos de stemming fueron evaluados sobre diferentes colecciones en varios idiomas con el objetivo de determinar su influencia en los resultados entregados por un SRI. Algunos autores [26] plantearon que aplicar la técnica de stemming a un corpus no mejora la recuperación de información para el inglés. Pero otros – en experiencias más contemporáneas – indican que sí hay mejoras sobre la recuperación en lengua inglesa en especial sobre el incremento de recall [28]. Popovic y Willett [42] han concluido que textos lematizados mejoran claramente la perfomance de un SRI que indexa documentos de lenguajes ricos morfologicamente.

Otras investigaciones sobre stemming y español [19] han demostrado que la normalización mejora los resultados de un SRI, en especial cuando las consultas son cortas. La técnica de los n-gramas se la desaconseja, debido a que los resultados obtenidos no alcanzan los que se obtienen sin aplicar ninguna técnica de normalización. Por otro lado, aquellos algoritmos que incluyen conocimiento lingüístico no son eficientes y obtienen calificaciones bajas.

En estudios propios [5] se ha demostrado que la adaptación del algoritmo de Porter para el castellano funciona correctamente, alcanzándose un factor de compresión del 76% aproximadamente, versus estudios similares con factor del 36% para el inglés. Por otro lado, el algoritmo contribuye al aumento de la frecuencia de ocurrencia de las palabras con poca presencia en el corpus.

Además de los métodos mencionados existen las técnicas basadas en el análisis lingüístico, que tratan de detectar familias de palabras a partir de utilizar la distancia semántica. Algunos estudios iniciales plantean que – en particular – para el español el desempeño de estos métodos es bastante bajo [19].

En la página de web http://www.snowball.tartarus.org/spanish/stemmer.html existe una implementación de un método de stemmer adaptado para el español, la cual está basada en la plataforma de procesamiento de lenguaje denominada Snowball.

3.4.4 – Selección de los términos a indexar

En esta fase se tiene por objetivo analizar cada documento a indexar a los efectos de extraer el juego de palabras (simples o compuestas) que representen de mejor forma su contenido. Como se mencionó anteriormente, los mejores términos candidatos a pertenecer a un vocabulario son aquellos que se hallan en la parte central de la gráfica de Luhn. Esto se debe – según estudios de Luhn [34] – a que la importancia de un término debería ser determinada en base a su frecuencia de aparición sobre el texto a procesar[13].

Por regla general, las palabras de alta frecuencia se eliminan con la asistencia de un diccionario de palabras vacías y las palabras de baja frecuencia – opcionalmente – también se eliminan. Los términos que superan el proceso de filtrado componen el vocabulario de la colección.

En esta sección se presentan dos recursos adicionales que permiten enriquecer el vocabulario y ganar una mayor performance en la tarea de recuperación: colocaciones y metadatos.

3.4.4.1 – Colocaciones

En general, existen en los documentos de texto grupos de dos o más palabras que – encontrándose asociadas – poseen un significado específico, propio del dominio temático en cuestión, por ejemplo: “sistema operativo”, “base de datos”, “information retrieval”, etc. A estos grupos se los conoce como *multipalabras*.

Un tipo particular de multipalabra son las colocaciones, definidas como la tendencia de ciertas palabras a co-ocurrir regularmente en un lenguaje dado. El estudio de las colocaciones se da en distintas áreas, por ejemplo: traducción automática, lexicografía, psicología y recuperación de información.

Las colocaciones se pueden detectar mediante la consulta en diccionarios de colocaciones o por extracción automática en base a distintos tipos de técnicas lingüísticas y estadísticas. Luego, pueden formar parte del vocabulario de la colección como una entrada más y – por ende – participa en la recuperación, recibiendo el mismo tratamiento que una palabra simple.

Si se analizan los tipos de colocaciones se encuentran: a) las unidades léxicas compuestas que funcionan como una sola palabra. Son un único concepto y no se pueden descomponer (por ejemplo: un sistema operativo no es algo "que es un sistema" y

[13] Luhn justificó el uso de frecuencias como métrica a partir del siguiente concepto *“.. un escritor repite normalmente ciertas palabras mientras avanza o varía sus argumentos y como él elabora un aspecto de un sujeto...”.*

"que es operativo"), y b) los sintagmas que sí se pueden descomponer (por ejemplo: computadora barata, grabadora de CD, etc.).

Patrones comunes de colocaciones son:

verbo	+	sustantivo	(volar un avión)
sustantivo	+	adjetivo	(placard marrón)
verbo	+	adverbio	(comer despacio)
adverbio	+	adjetivo	(totalmente diferente)
adjetivo	+	preposición	(similar a)
sustantivo	+	sustantivo	(libro de cocina)

A continuación se muestran las colocaciones más frecuentes extraídas del libro “Harry Potter y el Cáliz de Fuego” de J. K. Rowling.

1er término	2do término	Ocurrencias 1er término	Ocurrencias 2do término	Ocurrencias conjuntas
señor	weasley	642	380	191
señor	crouch	642	309	161
señora	weasley	198	380	119
madame	maxime	93	97	93
profesora	mcgonagall	156	94	87
rita	skeeter	97	88	81
tio	vernon	85	77	76
fred	george	189	158	76
voz	baja	383	96	76
ludo	bagman	70	224	51
ron	hermione	980	835	135
sala	comun	129	48	41
tia	petunia	41	40	39
marca	tenebrosa	53	41	37
viktor	krum	49	208	37
ojo	magico	73	59	34
profesora	trelawney	156	35	34
senor	diggory	642	110	42
elfos	domesticos	52	31	31

En el anexo 3 se presentan las colocaciones más frecuentes extraídas del corpus New York Times.

Existen diversas herramientas – derivadas de la teoría de la información y la estadística – para valorar si un grupo de palabras (n-grama) es una colocación o no (test de chi cuadrado, Z-score, likelihood ratios, etc.). Además, se cuenta con una métrica propuesta por Fano y estudiada por Church y Hanks [12] denominada Información Mutua (IM) que mide la fuerza de asociación entre dos palabras, es decir, la cantidad de información que la aparición de una palabra aporta sobre la aparición de la otra:

$$IM(A,B) = \log_2 \frac{P(A,B)}{P(A) \bullet P(B)}$$

Donde P(A,B) es la probabilidad que los términos A y B ocurran juntos
P(A) Probabilidad de ocurrencia del término A
P(B) Probabilidad de ocurrencia del término B

Esta medida estadística calcula la probabilidad de que las dos palabras (A y B) aparezcan juntas, calculando la probabilidad de que A y B aparezcan de forma independiente y después se comparan los dos valores. Si existe una asociación alta entre A y B, la probabilidad de que aparezcan juntas deberá ser mucho mayor que la de que aparezcan por separado. En caso de que los dos valores de frecuencia sean muy similares, la concurrencia de las dos palabras no suele considerarse significativa. A continuación se presenta un ejemplo de cómputo de información mutua sobre un corpus hipotético:

	B	No B
A	100	50
No A	25	1000

N = 100 + 50 + 25 + 1000 = 1175

P(A) = 125 / 1175 = 0,10
P(B) = 150 / 1175 = 0,12
P(A,B) = 100 / 1175 = 0,08

$$IM(A,B) = \log 2 \frac{0,08}{0,10 \bullet 0,12} = 2,73$$

En base a los experimentos de Church y Hanks se sugieren que los mejores conjuntos de colocaciones se obtienen con valores de IM > 3.

Para la extracción de colocaciones se recomienda realizar primero el análisis léxico, quitando palabras vacías y – de forma optativa – haber lematizado el vocabulario candidato. Para realizar esta tarea se puede recurrir al software desarrollado por Smadja [58] denominado Xtract. No obstante, a continuación se presenta un programa codificado en lenguaje Perl que extrae posibles colocaciones de un texto por el método de IM.

```
#!/usr/local/bin/perl
#
# Programa que extrae colocaciones de un archivo pasado como parametro
```

```
#
# uso perl busca-coloca.pl <nombre_archivo>
#

open(IN, $ARGV[0]) || die "Error al abrir el archivo\n";

$lastword = "";

while ($linea = <IN>)
{
  @words = split(' ', $linea);
  foreach $word (@words)
   {
    $word =~ tr/A-Z/a-z/;
    $lexicon{$word} = $lexicon{$word} + 1;
    $clave = $lastword."#".$word;
    $pairs{$clave} = $pairs{$clave} + 1;
    $q_palabras++;
    $numwords = $numwords + 1;
    $lastword = $word;
   }
}
close(IN);


# Listar colocaciones halladas halladas con MI >= 3

$q = $q_palabras * 2;
foreach $key (sort keys %pairs)
{
  ($alfa,$beta) = split(/#/, $key);
    $alfa_n = $lexicon{$alfa};
    $beta_n = $lexicon{$beta};
    if($alfa_n >= 1 && $beta_n >= 1)
    {
          $mi=log(($pairs{$key}/($q))/(($lexicon{$alfa}/($q))*($lexicon{$beta}/
($q))))/log(2);
      $dice = (2 * $pairs{$key}) / ( $alfa_n + $beta_n );
      if($mi >= 3)
      {
                                        print    "$key=$pairs{$key},$alfa=$alfa_n,
$beta=$beta_n,mi=$mi,dice=$dice\n";
      }
   }
}
```

3.4.4.2 – Metadatos

Literalmente, un metadato es “algo” que está por sobre los datos. Una definición comunmente utilizada determina "datos sobre datos", es decir que son objetos de información que describen o dicen algo sobre otro objeto de información.

Tradicionalmente, los metadatos son utilizados en ambientes de bibliotecas y sistemas documentales. Un caso común es asociar autor, título y fecha a una publicación particular con el objetivo de organizarlo bajo una estructura definida. Esto sirve para minimizar esfuerzos de organización y facilitar su mantenimiento.

Existen normas que definen diferentes estructuras de metadatos, las cuales persiguen objetivos diferentes, por ejemplo:

- Dublin Core (DC)[14]. Datos sobre páginas del espacio web.
- Consortium for the Interchange of Museum Information (CIMI)[15]. Informaciones sobre museos.
- Machine Readable Card (MARC)[16]. Catalogación bibliográfica.
- Federal Data Geographic Commitee (FGDC)[17]. Descripción de datos geo espaciales.

Estas normas definen lenguajes que especifican la sintaxis para generar estructuras y proveen especificaciones semánticas necesarias que explican el significado de las expresiones sintácticas.

Actualmente, a los efectos de ayudar a solucionar el problema de sobrecarga de información – derivada de la constante expansión del espacio web – se promueve el uso de la denominada web semántica. La cual tiene entre sus objetivos modificar la forma en que se presenta la información en el espacio web en pos de facilitar el procesamiento automático de la misma, y de esta forma establecer facilidades para lograr un factible procesamiento, integración y reutilización de la información contenida en tal espacio. Los metadatos juegan un rol importante en el área mencionada, debido a que proveen una categorización semántica de su contenido, permitiendo razonar de forma automática sobre la información.

En el siguiente ejemplo, se presentan los metadatos asociados a una página web.

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<meta name="Description" content="Revista Novatica de la Asociacion de Tecnicos de Informática, núm. 157, may.-jun. 2002, Recuperación de Información y la Web / Novatica, ATI's Journal and Magazine, #157, May-Jun. 2002 issue. Monograph: Information Retrieval and the Web">

<meta name="GENERATOR" content="Mozilla/4.72 [en] (Win95; I) [Netscape]">

<meta name="Author" content="Rafael Fernandez Calvo">

<meta name="KeyWords" content="ATI, España, Spain, Informática, Computer, Association, Publicaciones, Revista, Journal, Magazine, Novatica, Novatica, num. 157, mayo-junio, 2002 Recuperación de Información y la Web , #157, May-Jun. 2002 issue, Monograph, Information Retrieval, Web">
```

[14] The Dublin Core Metadata Initiative http://dublincore.org
[15] Consortium for the Interchange of Museum Information http://www.cimi.org
[16] Librería del Congreso Norteamericano http://www.loc.gov/marc/
[17] Federal Data Geographic Commitee http://clearinghouse1.fgdc.gov/

Un desafío para RI es la generación automática de metadatos a partir de un documento, ya sea que éstos se encuentren de forma explícita o implícita. Por ejemplo, el autor y el tema son datos que se encuentran en el documento, mientras que el idioma y el nivel de legibilidad están implícitos. Aunque de manera tradicional los metadatos pueden participar del proceso de especificación de una consulta, resulta también interesante contar con metadatos que auxilien al usuario durante la etapa de evaluación de la respuesta del SRI. Con éstos puede determinar de manera más simple la relevancia de cada ítem sin necesidad de descargar y leer el documento original. Si bien esto no es una mejora directa en la eficiencia del sistema de recuperación, resulta una mejora en la satisfacción del usuario en la evaluación de las prestaciones del SRI.

3.4.5 – Asignación de pesos

Para realizar operaciones de cálculo de similitud entre documentos y consultas, resulta necesario poder determinar la ponderación de cada término dentro de éstos, brindando a cada uno un peso o valor. El peso de cada término se basa en estudios de Luhn [34] y – generalmente – se encuentra relacionado con su frecuencia de aparición dentro de cada documento.

Por otro lado, esta tarea de ponderación o determinación de pesos puede ser considerada como de reducción de dimensionalidad. Esto se debe a que actúa como filtro para determinar cuáles son aquellos términos que sobrepasen un umbral de aceptación (palabras raras) o que no lo alcancen (palabras comunes) y – por lo tanto – no se tendrán en cuenta para representar el contenido del documento.

Inicialmente, el peso de un término t en un documento d se puede determinar por la aparición o no de un término o bien, se calcula a partir de obtener su frecuencia. Generalmente, esta frecuencia es normalizada para moderar el efecto de los términos que aparecen demasiado y compensar la longitud del documento.

3.4.5.1 – Método binario

Es el más simple de todos y consiste en asignar un 1 a la existencia de un término t y en un documento j o un 0 si no existe. Este método es utilizado – como veremos más adelante – por implementaciones del modelo de recuperación booleano y probabilístico.

	t_1	t_2	t_3	t_4
d_1	1	1	0	0
d_2	0	1	1	1
d_3	0	1	0	0

Una versión extendida consiste en almacenar en la matriz la cantidad de ocurrencias de cada término t en un documento d.

	t_1	t_2	t_3	t_4
d_1	6	1	2	3
d_2	0	2	0	0
d_3	0	1	0	1
d_4	2	5	1	0

d_5	0	0	4	2

A los efectos de comparar los pesos de los términos entre los documentos se pueden normalizar los valores entre 0 y 1. Para esto, se calcula el cociente entre la frecuencia y el largo del documento analizado (cantidad de palabras que posee), por ejemplo:

	t_1	**t_2**	**t_3**	**t_4**	**Largo**
d_1	2	5	2	3	12
d_2	0	2	0	0	2
d_3	0	1	0	2	2
d_4	2	0	1	0	3
d_5	0	0	4	2	6

Luego:

	t_1	**t_2**	**t_3**	**t_4**
d_1	0.16	5.00	2.00	3.00
d_2	0.00	2.00	0.00	0.00
d_3	0.00	0,33	0.00	0,66
d_4	0,66	0.00	0,33	0.00
d_5	0.00	0.00	0,66	0,33

Nótese en el cuadro anterior que el termino 1 para los documentos 1 y 4 posee la misma frecuencia (2), pero cuando se normalizaron los valores se asignó mayor peso al termino 1 en el documento 4. Esto ocurrió debido a que el documento 4 es de menor tamaño que el 1.

3.4.5.2 – Método TF*IDF

Una evolución de los métodos de ponderación presentados es el denominado TF*IDF (Term Frequency, Inverse Document Frequency), el cual plantea establecer una relación entre la frecuencia de un término dentro de un documento y su frecuencia en los documentos de la colección.

Básicamente, se utiliza la técnica de la "frecuencia del término por la frecuencia inversa", es decir, se obtiene la frecuencia pura del término t_i en el documento d_j (TF) y se lo multiplica por la inversa de la cantidad de documentos de la colección donde aparece t_i (IDF).

$$TF * IDF_{ij} = TF_{ij} \times \log_2 \frac{N}{n}$$

Donde,

- TF_{ij} Corresponde a la frecuencia pura del término t_i en el documento d_j (opcionalmente, se normaliza con la longitud de d_j ó la máxima frecuencia de un término en dicho documento).
- N Es el tamaño de la colección (cantidad de documentos).
- n Es la cantidad de documentos donde el término t_i aparece.

En el cálculo del IDF, los valores cercanos a cero indican que el término posee poco peso y por ende bajo valor de discriminación. Por otro lado, los valores positivos lejanos a cero indican que el término es poco frecuente y – por ende – resulta más adecuado para caracterizar a los documentos donde se encuentran. A continuación, se plantean algunos ejemplos:

N = 100 documentos

"la"	n = 100 documentos	$IDF_{(la)} = \log(100/100) = 0.00$
"casa"	n = 50 documentos	$IDF_{(casa)} = \log(100/50) = 1.00$
"Iran"	n = 15 documentos	$IDF_{(Iran)} = \log(100/15) = 2.73$
"Irak"	n = 10 documentos	$IDF_{(Irak)} = \log(100/10) = 3.32$
"ADA"	n = 1 documento	$IDF_{(ADA)} = \log(100/1) = 6.64$

Aquí se nota claramente que para cualquier término t_i, su peso será mayor mientras aparezca en menos documentos de la colección. Por otro lado, el valor de IDF varía entre 0 y el log(N).

Como se mencionó anteriormente, una variante a la fórmula de TF*IDF considera utilizar las frecuencias normalizadas a los efectos de relativizar los pesos en los documentos largos y cortos.

$$TF * IDF_{ij} = \frac{TF_{ij}}{l\arg o(d_j)} \times IDF_i$$

Donde,

largo(d_j) es la cantidad de términos del documentos d_j.

A continuación, se presenta un desarrollo completo del cálculo de pesos utilizando un corpus hipotético:

Doc_1: león león león
Doc_2: león león león zorro
Doc_3: león zorro nutria
Doc_4: león león león zorro zorro zorro
Doc_5: nutria

Representación matricial de las frecuencias de los términos:

	león	**zorro**	**nutria**
doc_1	3	0	0
doc_2	3	1	0
doc_3	1	1	1
doc_4	3	3	0
doc_5	0	0	1

Largo
3
4
3
6
1

Matriz de frecuencias normalizadas:

	león	**zorro**	**nutria**

doc$_1$	1.00	0.00	0.00
doc$_2$	0.75	0.25	0.00
doc$_3$	0.33	0.33	0.33
doc$_4$	0.50	0,50	0.00
doc$_5$	0.00	0.00	1.00

El cálculo de IDF al vocabulario del corpus resulta:

	N	**n**	**IDF**
león	5	4	0,32
nutria	5	3	0,73
zorro	5	2	1,32

Luego, el cálculo del TF*IDF normalizado:

	león	**zorro**	**nutria**
doc$_1$	0,32	0.00	0.00
doc$_2$	0.24	0.33	0.00
doc$_3$	0.10	0.43	0.24
doc$_4$	0.16	0,66	0.00
doc$_5$	0.00	0.00	0,73

Un abuso sobre la frecuencia de un término puede producir su peso aumente artificialmente y – luego – que tales documentos (generalmente, no relevantes a la consulta) se rankeen en los primeros lugares produciendo ruido documental. Un caso típico en la Web son las páginas de publicidad o sitios de subastas.

Un extracto de una página que favorece ciertas frases:

Silvina Luna Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar **Silvina Luna** Maria Eugenia Rito Leticia Bredice Carla Petersen Luciana Salazar

3.4.5.3 – Valor de discriminación de un término

Es un proceso que tiene por finalidad detectar aquellos términos del vocabulario de una colección que identifiquen de mejor manera los documentos [50]. En otras palabras, se clasifican los términos de un texto según la capacidad de éstos para discriminar unos

documentos de otros en un corpus. La determinación del valor de discriminación de un término t (VDT) en una colección C se calcula de la siguiente forma:

VDT(t_i) = Semejanza promedio colección sin t_i – Semejanza promedio de la colección

Primero, se calcula la semejanza de cada documento con respecto a los demás de la colección utilizando alguna fórmula conocida como la del coseno o coeficiente de Dice y se promedian los resultados obtenidos. Luego, se repite la misma operación para cada término del vocabulario pero sin considerarlo en el cálculo. De esta manera, se obtienen tantos promedios como términos haya en el vocabulario.

A continuación, se computa la diferencia de cada promedio obtenido menos el promedio de la colección. Si esta diferencia para un término t es negativa, se considera que t no es un buen discriminador y debería ser eliminado del vocabulario. Para valores cercanos a cero, t es bastante frecuente en los documentos de la colección y posee un valor de discriminación bajo. En cambio, para valores grandes superiores a cero, t es un buen término discriminador de documentos.

A los efectos de ilustrar esta métrica, se presenta un ejemplo para un corpus de 4 documentos y un vocabulario de 5 términos. Se calculó la semejanza promedio de la colección a partir de las aparición de los términos (aunque se puede utilizar la frecuencia) y el cálculo de similitud por el coeficiente de Dice, definido como:

Coeficiente de Dice: $$\frac{2 \bullet \sum_{j=1}^{t} w_{dij} \bullet w_{qj}}{\sum_{j=1}^{t} w_{dij}^2 + \sum_{j=1}^{t} w_{qj}^2}$$

Luego,

corpus	t_a	t_b	t_c	t_d	t_e	**largo**
d1	1		1		1	3
d2	1	1		1		3
d3	1		1			2
d4		1		1		2

Cómputo de semejanza Corpus completo		
Dice(d1, d2)	2*1/(3+3)	0,333
Dice(d1, d3)	2*2/(3+2)	0,800
Dice(d1, d4)	2*0/(3+2)	0,000
Dice(d2, d3)	2*1/(3+2)	0,400
Dice(d2, d4)	2*2/(3+2)	0,800
Dice(d3, d4)	2*0/(2+2)	0,000
Semejanza promedio		0.389

En el segundo paso, se calculó la semejanza sin el término T_a:

corpus	t_a	t_b	t_c	t_d	t_e	**largo**
d1			1		1	2
d2		1		1		2
d3			1			1
d4		1		1		2

Cómputo de semejanza sin el término "T_a"		
Dice(d1, d2)	2*0/(2+2)	0,000
Dice(d1, d3)	2*1/(2+1)	0,667
Dice(d1, d4)	2*0/(2+2)	0,000
Dice(d2, d3)	2*0/(2+1)	0,000
Dice(d2, d4)	2*2/(2+2)	1,000

Dice(d3, d4)	2*0/(1+2)	0,000
Semejanza promedio		0.278

$VDT(T_a) = 0.278 – 0.389 = – 0.111$

Como $VDT(T_a)$ es negativo, indica que T_a no es un buen discriminador.

En el siguiente paso, se calculó la semejanza sin el término T_b:

corpus	t_a	t_b	t_c	t_d	t_e	**largo**
d1	1		1			2
d2	1	1		1		3
d3	1		1			2
d4		1		1		2

Cómputo de semejanza sin el término "T_e"		
Dice(d1, d2)	2*1/(2+3)	0,400
Dice(d1, d3)	2*2/(2+2)	0,667
Dice(d1, d4)	2*0/(2+2)	0,000
Dice(d2, d3)	2*1/(3+2)	0,400
Dice(d2, d4)	2*2/(3+2)	0,800
Dice(d3, d4)	2*0/(2+2)	0,000
Semejanza promedio		0,378

$VDT(T_e) = 0.378 – 0.389 = – 0.011$

Como $VDT(T_a)$ es negativo pero cercano a cero, indica que T_b tiene una bajo poder de discriminación.

El paso posterior corresponde al cálculo sin T_c:

corpus	t_a	t_b	t_c	t_d	t_e	**largo**
d1	1				1	2
d2	1	1		1		3
d3	1					1
d4		1		1		2

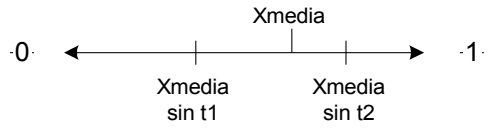
Cómputo de semejanza sin el término "T_c"		
Dice(d1, d2)	2*1/(2+3)	0,400
Dice(d1, d3)	2*1/(2+1)	0,667
Dice(d1, d4)	2*0/(2+2)	0,000
Dice(d2, d3)	2*1/(3+1)	0,500
Dice(d2, d4)	2*2/(3+2)	0,800
Dice(d3, d4)	2*0/(1+2)	0,000
Semejanza promedio		0.394

$VDT(T_c) = 0.394 – 0.389 = 0.005$

Como $VDT(T_c)$ es positivo T_c es un buen discriminador.

El problema asociado a este método de discriminación de términos es su costo computacional de orden $O(n^2 * (l + 1))$, donde n es la cantidad de documentos en la colección y l el largo del vocabulario. Esto de debe a que requiere computar la semejanza de todos los documentos entre si l + 1 veces.

Conceptualmente, el valor de discriminación de un término puede verse representado en el siguiente gráfico:



El valor promedio de la semejanza de un corpus está entre cero y uno. Cuando éste se acerca a cero – en general – existe una gran diferencia entre todos los documentos de la colección. En cambio, cuando se acerca a uno existe una importante semejanza. A partir de tal observación se prueba como varía la semejanza promedio eliminando un término del vocabulario por vez. En caso que la diferencia sea negativa (Xmedia sin t_1 – Xmedia) se asume que el término en cuestión no ayuda a diferenciar documentos. Por otro lado, cuanto más se aleje de la media (Xmedia sin t_1 – Xmedia) hacía la recta de los positivos, mejor es su poder de discriminación.

Una alternativa para reducir el costo computacional consiste en utilizar una técnica que define un documento centroide. Éste es un documento virtual formado por el promedio de todos los documentos de la colección, donde el peso de cada termino T_i es el promedio de los T_i de todos los documentos. Este documento se utiliza como referencia para los cálculos posteriores [39].

3.5 – Actividades

1) Utilice tres documentos en distintas lenguas de al menos 100 KB cada uno. Escriba un programa que obtenga las frecuencias asociadas a los términos y analice visualmente el cumplimiento de la ley de Zipf mediante un gráfico. Obtenga las listas de palabras vacías para las lenguas seleccionadas.

2) Se disponde de una colección con 1.000.000 de términos, de los cuales 410.000 son únicas. Se debe decidir cuáles se incluyen en el vocabulario que se utilizará para la construcción de las estructuras de datos. Para ello se toma el criterio de descartar aquellas que ocurren 3 o menos veces y las 10 más frecuente. Suponiendo que el corpus se ajusta a la ley de Zipf, estime la longitud del vocabulario.

3) Existe un requerimiento consistente en indexar un corpus cuyo tamaño es de 800.000 palabras. Del análisis de pequeñas muestras del corpus se obtuvo que para 10.000 palabras, el tamaño del vocabulario es de aproximadamente 25.000 términos únicos y para 50.000 palabras de aproximadamente 78.000. ¿Cómo puede estimar el tamaño final del vocabulario del corpus completo? ¿Qué cantidad de memoria se requiere para almacenarlo? (va a necesitar información extra). ¿Qué sucedería en caso que el corpus tenga 1.000.000.000 de palabras?

4) Tome el algorimo de Porter y lematize los siguientes términos: query, queries, asked, friends, friendship, stemmer, stemming. Luego pruebe con: spicy, spice, spices, beautiful, beauty. Luego reponda:

   a) ¿Cúal es la función del método de lematización?
   b) ¿Es útil el método? ¿Por qué?
   c) ¿Adolece de algún problema visible?

5) Implemente el algoritmo de S-Stemmer en lenguaje Perl para la lengua inglesa y para el español.

6) Calcule el peso según el método TF*IDF del siguiente extracto de un vocabulario de un corpus de 965 documentos.

Término	Frecuencia del término TF(t)	Cantidad de documentos donde está el término	Peso TFIDF
abril	Doc1: 1	324	
	Doc2: 2		
arabia	Doc4: 4	87	
	Doc5: 1		
	Doc7: 2		
corte	Doc2: 1	229	
	Doc3: 3		
demarcar	Doc5: 1	88	
demoler	Doc6: 1	2	
	Doc9: 1		
en	Doc1: 15	892	
dilemma	Doc5: 4		

Capítulo 4

Modelos de recuperación de información

La recuperación de información trata de encontrar documentos relevantes de acuerdo a una necesidad de información, expresada como una consulta, de un usuario. Como se ha mencionado, esta tarea es imprecisa debido a las simplificaciones que se realizan en todo el proceso, por ejemplo, el mapeo de la necesidad de información del usuario en un *query* o la visión lógica de un documento como un conjunto de palabras (generalmente denominada "bolsa de palabras").

Cuando un SRI recibe una consulta realiza un proceso de búsqueda sobre sus índices y – de alguna manera – calcula una "puntuación" para cada documento. Solo a aquellos que superen dicho puntaje se los considera relevantes. Luego, este valor permite determinar un orden o ranking de la respuesta para el usuario. Ahora bien, ¿Cómo se determina la "puntuación"? Existen varias aproximaciones para realizar esta tarea basadas en diferentes conceptos y suposiciones que determinan "modelos de recuperación de información" distintos.

Los modelos de recuperación de información definen cuáles son las premisas que se tienen en cuenta para determinar si un documento es relevante o no a una necesidad de información. En otras palabras, un modelo determina cómo se realizará la comparación entre consulta y documentos para calcular una medida de similitud que permita determinar la relevancia (o no) y el ranking.

Para aproximar una solución al problema planteado – inicialmente – se presentan dos estrategias triviales que la resuelven en parte, aunque no son eficientes en costo computacional:

a) Búsqueda por texto libre
b) Búsqueda ingenua

El primero de los casos es el más simple e ineficiente. Cuando un usuario ingresa una consulta, ésta se resuelve analizando cada documento del corpus de forma secuencial a un costo O(n). Simplemente, se busca que el patrón ingresado se encuentre en el documento y – si existe coincidencia – éste pertenece al conjunto solución. En esta estrategia no existen estructuras de datos que auxilien a la consulta y el conjunto solución no se rankea de manera alguna. Se justifica su utilización solamente en colecciones pequeñas y dinámicas.

El segundo de los casos – la búsqueda ingenua – es una optimización al modelo anterior que permite rankear los documentos de la respuesta. Suponga que se tienen las siguientes estructuras de datos:

Consulta = $\{q_1, q_2, .., q_n\}$

$Doc_i = \{(t_1,p_1), (t_2, p_2), ..., (t_m, p_m)\}$ (Los términos se encuentran en orden alfabético)

Donde *q* es un término de una consulta
t un término existente en un documento
p peso asociado al término, que indica su importancia

El algoritmo de recuperación recorre todos los documentos y – por cada uno – calcula la sumatoria de pesos de los términos de la consulta que se encuentren en cada documento. Con esta operación se obtiene un valor de importancia del documento. Una vez que se procesó todo el corpus se muestra el conjunto solución, el cual está formado por todos aquellos documentos cuyo valor de importancia es mayor que cero, ordenados descendentemente. A continuación, se muestra un ejemplo de esta estrategia de búsqueda:

Doc_1 = {(casa, 0.25), (monitor, 0.33), (termo, 0.42), (ventana, 0.61) }
Doc_2 = {(ajo, 0.15), (gato, 0.29), (pelo, 0.21), (vaso, 0.52) , (yeso, 0.46)}
Doc_3 = {(casa, 0.33), (gato, 0.11), (ventana, 0.92), (yeso, 0.61) }

Consulta = {“casa”, “termo”, “yeso”)

Cálculo de relevancia

Doc_1 = 0.25 + 0.42 = 0.67
Doc_2 = 0.46 = 0.46
Doc_3 = 0.33 + 0.61 = 0.94

Por lo tanto, la respuesta resulta: Doc_3, Doc_1, Doc_2

Desde la aparición del área de RI, se han propuesto y evaluado muchos modelos de fundamentos completamente diferentes. Generalmente, reciben el nombre de modelos de *matching* aproximado (*approximate matching models*) ya que – en general – utilizan la frecuencia de aparición y la distribución de los términos dentro de cada documento para calcular el puntaje.

Ninguno de los modelos ha resuelto el problema central de la RI de manera definitiva, sino que – debido a las premisas en que se basan – cada uno presenta ventajas y desventajas.

Baeza-Yates [2] brindó una caracterización formal de los modelos de recuperación de información:

Definición: Un modelo de RI es una cuádrupla [$\boldsymbol{D, Q, F, R(q_i, d_j)}$], donde:

D es el conjunto de las representaciones lógicas de los documentos de la colección.

Q es el conjunto de las representaciones lógicas de las consultas (necesidades de información del usuario).

F es un marco para modelar los documentos, las consultas y las relaciones entre ambos.

$R(q_i, d_j)$] es la función de ranking, la cual asocia un número real con la representación de una consulta $\mathbf{q_i}$ ($\mathbf{q_i} \in \mathbf{Q}$) y la representación de un documento $\mathbf{d_j}$ ($\mathbf{d_j} \in \mathbf{D}$).

A partir de esta formalización, quedan claramente expresados los componentes de un modelo de RI y la relación existente entre éstos.

4.1 – Modelos clásicos

Tradicionalmente, existen en el área de RI tres modelos clásicos para tratar de brindar soluciones a las necesidades de información de los usuarios:

a) El booleano, basado en la teoría de conjuntos y álgebra de Boole.

b) El vectorial, que trata a las consultas y a los documentos como vectores en un hiperespacio, operando mediante el álgebra de vectores.

c) El probabilístico, que se basa en procesos estocásticos, operaciones de la teoría de la probabilidad y el teorema de Bayes.

De acuerdo a lo planteado por Baeza-Yates, los fundamentos de cada modelo constituyen el marco conceptual (F) que define cómo se representan y relacionan los documentos D y las consultas Q. Además, brindan la intuición para la construcción de la función de ranking $R(q_i, d_j)$.

En este capitulo, se presentan los modelos booleano y vectorial, donde este último es el más difundido y utilizado por la comunidad de RI. Si bien estos modelos son la base para extensiones y nuevos puntos de vista, no es nuestro objetivo tratarlos en detalle.

4.1.1 – El Modelo Booleano

Como se mencionó, está basado en la teoría de conjuntos y el álgebra de Boole. Su uso es de larga data y ha gozado de popularidad en ambientes comerciales. En este modelo, cada documento se representa por un conjunto de términos, donde cada uno se trata como una variable booleana que se instancia en verdadero si el término está presente en el documento y en falso por lo contrario. Hay un esquema de pesos binarios asociados a los términos, lo cual implica que no interesa si el término es de alta o baja frecuencia, solo importa su ausencia o presencia.

Formalmente, el modelo booleano [***D, Q, F, $R(q_i, d_j)$***], se define como:

D Conjunto de términos presentes en los documentos.
Q Expresión booleana formada por términos y operadores (OR, AND, NOT).
F Álgebra booleana sobre los conjuntos de términos y documentos.
$R(q_i, d_j)$ Un documento se considera relevante a una consulta si satisface la expresión de consulta. No existe ranking alguno.

Los pesos de los términos están en el dominio 0/1. Considérese el siguiente ejemplo:

Términos, t = {t1, t2, t3}

Documentos,A = (1, 1, 1)
B = (1, 1, 0)
C = (1, 0, 0)

Consulta, q = {t1 /\ (¬ t2 V t3)}

T1= {A, B, C}
¬ T2= {C}
T3= {A}

{A,B,C} AND ({C} OR {A})
{A,B,C} AND ({C, A}) Respuesta {C, A}

A continuación, se muestra un corpus de cuatro documentos al cual se le eliminan las palabras comunes o de alta frecuencia y se lo representa en una matriz booleana documentos–términos:

Doc_1 = “El sol sale para todos” => Doc_1’ = {sol, sale, todos}
Doc_2 = ”Estoy en la vereda del sol” => Doc_2’ = {estoy, vereda, sol}
Doc_3 = ”Todos ven el sol salir” => Doc_3’ = {todos ven sol salir}
Doc_4 = ”La vereda sale cara” => Doc_4’ = {vereda, sale, cara}

	sol	sale	todos	estoy	vereda	ven	salir	cara
Doc_1’	1	1	1	0	0	0	0	0
Doc_2’	1	0	0	1	1	0	0	0
Doc_3’	1	0	1	0	0	1	1	0
Doc_4’	0	1	0	0	1	0	0	1

Es común que la longitud del vocabulario sea extensa en colecciones importantes. Tal situación atenta contra los tiempos de recuperación debido a la cantidad de datos a manejar. Existen técnicas de poda del vocabulario (conocidas también como reducción de dimensionalidad) que eliminan aquellos términos muy frecuentes o que raras veces aparecen, las cuales se presentaron en el capítulo tres.

En general, los usuarios en la consulta utilizan operadores booleanos clásicos: AND (/\), OR (V), NOT (¬), ya sea tipeándolos expresamente u operando en lenguaje natural, donde el sistema luego convertirá tales expresiones en consultas booleanas.

Las búsquedas se dividen en base a los términos que las componen. Primero se recuperan los conjuntos de documentos asociados a cada término. Luego, tales conjuntos son combinados de acuerdo a los operadores booleanos para obtener un único conjunto solución.

Veamos otro ejemplo:

Términos (Vocabulario):

$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$

Documentos :

$Doc_1 = \{t_1, t_2, t_3, t_4, t_5\}$
$Doc_2 = \{t_1, t_2, t_3, t_4\}$
$Doc_3 = \{t_2, t_4, t_6, t_8\}$
$Doc_4 = \{t_1, t_3, t_5, t_7\}$
$Doc_5 = \{t_4, t_5, t_6, t_7, t_8\}$
$Doc_6 = \{t_1, t_2, t_3, t_4\}$

Consulta = t_1 **AND** (t_2 **OR** (**NOT** t_3))

Luego,

$S_1 = \{doc_1, doc_2, doc_4, doc_6\}$
$S_2 = \{doc_1, doc_2, doc_3, doc_6\}$
$S_3 = \{doc_3, doc_5\}$

$\{doc_1, doc_2, doc_4, doc_6\}$ **AND** $\{doc_1, doc_2, doc_3, doc_5, doc_6\}$

Solución:

$S = \{doc_1, doc_2, doc_6\}$

Aunque el modelo es simple, expresar una necesidad de información mediante operadores booleanos no es una tarea trivial. Considérese el siguiente escenario:

Necesidad de información = “Información sobre la reproducción de la ballena franca austral y su habitat”

La consulta expresada en el modelo booleano se puede plantear como:

Consulta = ”ballena **AND** franca **AND** austral AND (‘Península Valdes‘ **OR** ‘Puerto Madryn‘) **AND** reproducción ”

Sin embargo, queda claro que para expresar la consulta se debe recurrir a información extra que el usuario conoce, aunque no siempre. Ciertas veces, usuarios inexpertos cometen errores al expresar su consulta, tal como el siguiente caso:

Necesidad de información = “Obtener información sobre Península de Valdes y Puerto Madryn”

Consulta (erronea) = ” ‘Peninsula Valdes‘ **AND** ‘Puerto Madryn‘”

En este ejemplo se construye erróneamente la consulta dado que se restringen el conjunto de documentos solución por no haber utilizado el operador OR.

Por otro lado, no es común que los usuarios recuerden las reglas de precedencia para los operadores booleanos (NOT, AND y OR) y cuando plantean una consulta del

tipo: “comida **OR** perros **AND** gatos” debe procesarse como “comida **OR** (perros **AND** gatos)”. Este error es común en usuarios nóveles y – en tal caso – perderán aquellos documentos relevantes que hablen acerca de uno de los temas y no del otro.

Como una característica del modelo booleano se debe resaltar que el conjunto respuesta siempre está integrado por documentos que satisfagan absolutamente la consulta y no hay posibilidad de realizar *ranking* alguno. Se asume que cada documento recuperado es de la misma importancia que sus pares. Esto sucede debido a que no es posible determinar la relevancia relativa de cada documento respuesta a la consulta expresada.

Uno de los puntos débiles del modelo booleano está dado por que en ciertas situaciones puede brindar resultados no óptimos. Suponga que se tiene la siguiente consulta:

Consulta = “A **OR** B **OR** C **OR** D”

Como respuesta, cualquier documento que contenga los cuatro términos o solo uno será presentado con la misma importancia. Por otro lado, si se tiene una consulta como:

Consulta = “A **AND** B **AND** C **AND** D”

Aquellos documentos que contengan los términos D, C y B (pero no A) serán considerados como no relevantes, de igual forma que aquellos documentos que no contenga ninguno de éstos.

Como se ha mencionado, en este modelo están claros sus fundamentos y es simple de comprender y operar. Debido a que no posibilita ranking de documentos, las expresiones de búsqueda pueden ser demasiado restrictivas y se recuperarán pocos documentos, o bien, poco restrictivas y se recuperarán muchos documentos. No obstante, este modelo ha sido utilizado en diversos sistemas y en los primero motores de búsqueda sobre el espacio web. Un SRI comercial denominado Knosys[18] implementa el modelo booleano.

Para aumentar las prestaciones del modelo, se ha enriquecido el lenguaje de consulta. Según Salton [50] se pueden distinguir los siguientes operadores complementarios a los booleanos:

a) Operadores posicionales

El uso de estos operadores potencia las capacidades de un sistema de recuperación basado en el modelo booleano. Consideran el valor del término dentro de su contexto. Los operadores posicionales se dividen en dos clases:

Posicionales absolutos: Posibilitan buscar un término en un lugar determinado del documento. Trabajan como operadores de campo, permitiendo que el usuario determine sobre que campo/s se debe restringir la búsqueda.

[18] Ver http:// www.knosys.net/

Por ejemplo, si se desea buscar libros de “Jorge Luis Borges” podría utilizarse un operador posicional restringiendo la selección al atributo “Autor” para el valor de búsqueda "Borges".

Buscar documentos donde Autor = “Borges”

Nótese que aquí resulta necesario contar con ciertos elementos estructurales definidos, como el metadato Autor.

Posicionales relativos o de proximidad: Posibilitan establecer la posición o separación máxima de un término respecto a otro dado. Se basan en el prinicipio que si dos términos ocurren en un mismo contexto puede haber una relación significativa.

Recuperar todos aquellos documentos donde los términos “Bush” y “guerra” estén en el mismo párrafo.

Buscar documentos que contengan el término “documento” a menos de 5 palabras del término “SGML”.

b) Operadores de comparación

Determinan un rango de búsqueda, fijando límites para la consulta. Tales límites pueden ser tanto numéricos como alfabéticos. Los operadores corresponden adquieren formas del tipo "mayor que", "menor o igual que".

Halle documentos donde año_publicación >= 1973

Como se puede deducir del ejemplo anterior, tales operados requieren de atributos o metadatos previamente definidos.

c) Operadores de truncamiento

En ciertas ocasiones, es necesario buscar por una familia de términos relacionados morfológicamente. Para facilitar este tipo de búsquedas se han introducido operadores de truncamiento, los que definen máscaras de consulta. Se trata de operadores que – normalmente – se los denota con símbolos como *, ? (comodines), y cuya presencia puede sustituir a un carácter o a un conjunto de éstos. Por ejemplo:

c_1 = sistem*	Coincidirán todos aquellos documentos que contengan términos que comiencen con la cadena “sistem” (sistema, sistemas, sistemática, etc.)
c_1 = carl?s	Coincidirán todos aquellos documentos que contengan términos que comiencen con la cadena “carl”, continúe con un único caracter cualquiera y finalice con el caracter “s” (carlos, carlas)

c_2 = "bibliote*" **AND** "clasific??" ¿Qué documentos coinciden con esta expresión?

Estos operadores se pueden combinar para construir consultas de mayor complejidad, que expresen más acabadamente las necesidades de los usuarios. El tipo de operadores a implementar, como es su sintaxis y su prioridad de evaluación dependerá de cada sistema de recuperación de información que los utilice.

Los beneficios de uso del modelo booleano están dados por que es simple de utilizar para ciertos grupos de usuarios. Es el más difundido y popular y computacionalmente posee una buena relación costo/beneficio. Por otro lado, el modelo booleano es de fácil implementación, las consultas son de rápido procesamiento y necesita relativamente poco espacio de almacenamiento. A los efectos que los usuarios logren interactuar con eficiencia, frente a un SRI booleano, deben comprender los operadores y utilizarlos de manera completa.

4.1.2 – Modelo booleano extendido

El modelo booleano extendido trata de contemplar la problemática de establecer un ranking sobre el conjunto de documentos respuesta a una pregunta. Utiliza como pesos la frecuencia de los términos en cada documento y computa la relevancia en base a una función de ranking. A continuación se puede observar el esquema utilizado por el motor de consulta de Sony [60] para ponderar las respuestas:

x **AND** y -> tf(x) * tf (y)
x **OR** y -> tf (x) + tf (y)
NOT x -> 0 si tf (x) > 0 ó 1 si tf (x) == 0

A continuación, se presenta un ejemplo del uso de estos operadores, supóngase una colección cualquiera:

tf	**t_1**	**t_2**	**t_3**	**t_4**
d_1	12	0	1	0
d_2	0	0	0	1
d_3	3	3	2	1
d_4	1	2	3	4
d_5	0	2	2	0
d_6	5	5	1	1

Matriz frecuencia de términos (tf)

1) t_1 **AND** t_2

d_3	(3*3)	9
d_4	(1*2)	2
d_6	(5*5)	25

Ranking (t_1 **AND** t_2) = d_6, d_3, d_4

2) t_1 **OR** t_2

d_1	(12+0)	12
d_3	(3+3)	6
d_4	(1+2)	3
d_5	(0+2)	2
d_6	(5+5)	10

Ranking (t_1 **OR** t_2) = d_1, d_6, d_3, d_4, d_5

3) t_1 **AND NOT** t_2

d_1	(12*1)	12
d_3	(3*0)	0
d_4	(1*0)	0
d_6	(5*0)	0

Ranking (t_1 **AND NOT** t_2) = d_1

4) (t_1 **OR** t_2) **AND** t_3

d_1	((12+0)*1)	12
d_3	((3+3)*2)	12
d_4	((1+2)*3)	9
d_5	((0+2)*2)	4
d_6	((5+5)*1)	10

Ranking ((t_1 **OR** t_2) **AND** t_3) = d_1, d_2, d_6, d_4, d_5

Como alternativa al modelo booleano puro se ha propuesto el **modelo de lógica borrosa**, basado en la teoría de conjuntos borrosos [44]. Este permite precisar niveles de pertenencia a conjuntos, dejando de operar dicotómicamente. Sobre los documentos se asocia un peso para cada término indexado (métrica que determina el grado en que dicho documento está caracterizado por tal término). Se redefinen los operadores lógicos a fin de que operen con esta información al momento de computar la similitud entre la consulta y el documento.

4.1.3 – Modelo Vectorial

El modelo vectorial – también denominado modelo de espacio vectorial – fue desarrollado por Salton como parte del proyecto del sistema SMART [49]. Se basa en cálculos que permiten introducir un orden (ranking) en los documentos recuperados en función de su relevancia respecto de la consulta. Plantea la necesidad de utilizar una función de similitud entre el documento y la consulta [50] [52] [53].

En el modelo vectorial, cada documento de la colección está representado por un vector t-dimensional, donde t es la cardinalidad del conjunto de términos indexados que representan a un corpus de documentos. Cada elemento del vector corresponde al peso del término asociado a esa dimensión.

En un esquema binario – el más simple pero también el más ineficaz – se asignan a los elementos del vector un 1 si la palabra forma parte de documento y un 0 en caso contrario. No obstante, es de uso común que los pesos asociados a los términos indiquen una medida de relevancia basada en un cálculo de frecuencias. Generalmente, se utiliza la métrica de ponderación TF*IDF, la cual ha sido explicada en el capítulo tres.

En el momento de la recuperación, las consultas se describen de la misma forma, generando un vector consulta. Luego, se procede a realizar los cálculos algebraicos para determinar la semejanza (Por ejemplo, mediante el producto escalar) entre el vector consulta y cada uno de los vectores que representan a los documentos del corpus. Finalmente, se realiza el ranking, de forma descendente por el valor de similitud calculado y se presenta la respuesta al usuario.

Formalmente, el modelo vectorial [***D, Q, F, R(q$_i$, d$_j$)***], se define como:

D Conjunto de términos presentes en los documentos.
Q Conjunto de términos que forman la consulta
F Representación de D y Q como vectores en un espacio t-dimensional y álgebra de vectores
R(q$_i$, d$_j$) Se considera la similitud entre un documento y una consulta de acuerdo a una medida de semejanza, por ejemplo, el coseno del ángulo que forman los vectores. Existe matching aproximado y ranking por similitud.

En el gráfico 1 se presenta un espacio de 2 dimensiones, donde cada una corresponde a un término diferente (A y B). Sobre éste se representan los vectores correspondientes a los documentos (d_1, d_2, d_3, d_4 y d_5) y a la consulta (q) de acuerdo a los términos en éstos.

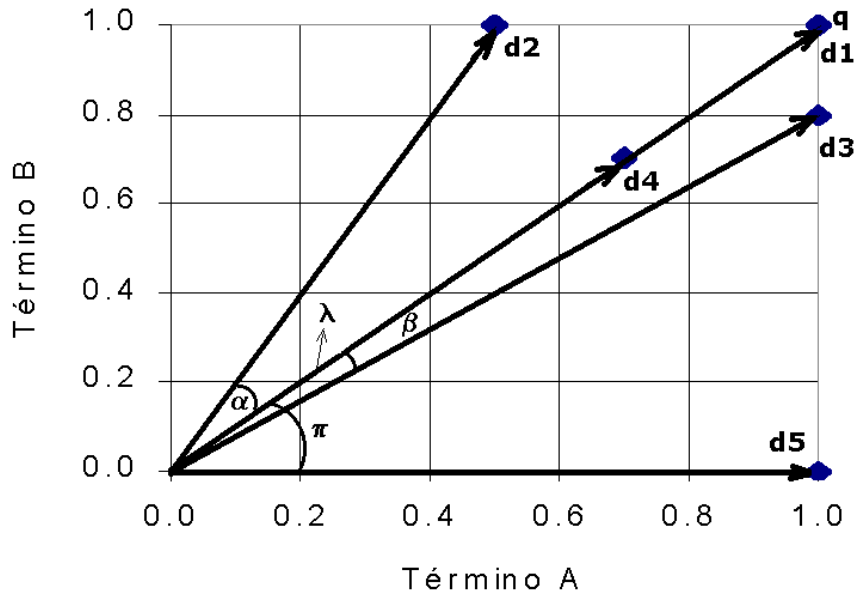


Gráfico 1 – Representación de documentos y consulta en un espacio bidimensional

De manera similar, si se tiene un vocabulario de solo tres términos, se lo puede representar en un espacio tridimensional. Suponga el siguiente ejemplo de un corpus de tres documentos y tres términos:

Documentos Términos	***A***	***B***	***C***
d_1	1.0	1.0	1.0
d_2	0.5	1.0	0.5
d_3	1.0	0.8	1.0
q	1.0	1.0	1.0

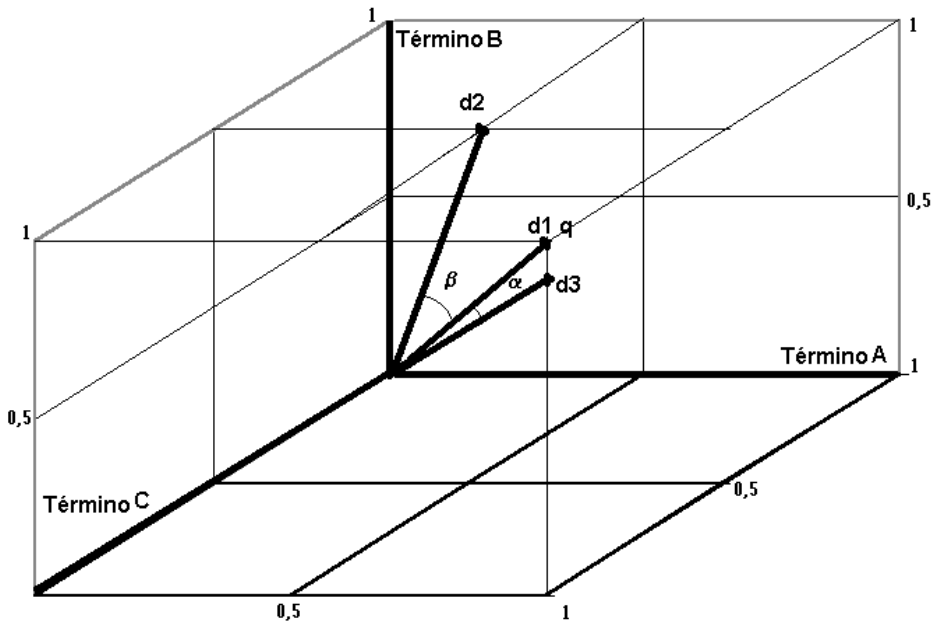


Gráfico 2 – Representación de documentos y consulta en un espacio tridimensional

Tanto la asignación de los pesos a los términos en los documentos [Salton, 1988] como el cálculo de similitud se puede realizar de diferentes maneras. En el primer caso, teniendo en consideración la frecuencia de aparición de la palabra. Se supone que el peso o valor de importancia de un término en un documento es inversamente proporcional a su frecuencia en el corpus o colección de documentos, y a su vez es directamente proporcional a su frecuencia en el documento. El cálculo ampliamente utilizado para el cálculo de los pesos de los términos en los documentos se basa en TF*IDF , definido como:

$$w_{ij} = f_{ij} * \log \frac{N}{n_i} \qquad \text{donde:} \qquad f_{ij} = \frac{freq_{ij}}{max_l freq_{l,j}}$$

Para el segundo caso, existen varias formas de calcular la similitud entre dos vectores. La similitud por medida del coseno del ángulo que forman el vector documento y el vector consulta, es la más popular de las métricas de semejanza. Se define como:

$$sim(Q, D_i) = \frac{\sum_{j=1}^{t} w_{q_j} \times w_{d_{ij}}}{\sqrt{\sum_{j=1}^{t} (w_{q_j})^2 \times \sum_{j=1}^{t} (w_{d_{ij}})^2}}$$

donde:

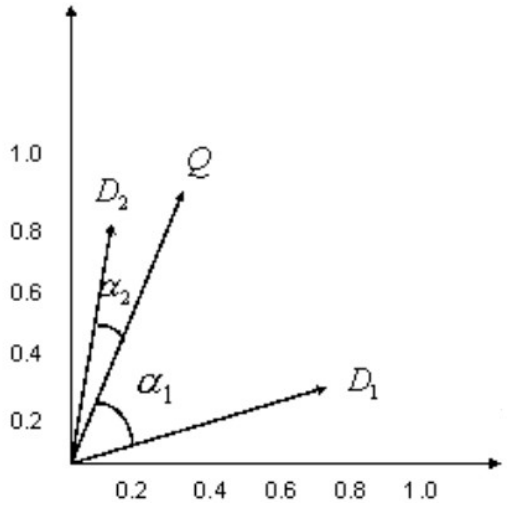
$D_i = w_{di1}, w_{di2}, w_{di3}, \ldots, w_{dit}$ Vector que representa el documento
$Q = w_{q1}, w_{q2}, w_{q3}, \ldots, w_{qt}$ Vector que representa la consulta

t : Número de términos en la colección
w_{dij} : Peso del término j en el documento i

Por ejemplo:

D_1 = (0.8, 0.3)
D_2 = (0.2, 0.8)
Q = (0.4, 0.8)

Su presentación gráfica:



La similitud entre los cada documento y la consulta resulta:

Cos α_1 = 0.74

$$sim(Q, D_2) = \frac{(0.4 \times 0.8) + (0.8 \times 0.3)}{\sqrt{(0.4)^2 + (0.8)^2 \times (0.8)^2 + (0.3)^2}} = \frac{0.56}{\sqrt{0.58}} = 0.74$$

Cos α_2 = 0.98

$$sim(Q, D_2) = \frac{(0.4 \times 0.2) + (0.8 \times 0.7)}{\sqrt{(0.4)^2 + (0.8)^2 \times (0.2)^2 + (0.7)^2}} = \frac{0.64}{\sqrt{0.42}} = 0.98$$

Una suposición que se realiza al adoptar esta representación espacial de t-dimensiones es que los términos ocurren de manera independiente dentro de los documentos. Esto se expresa en el hecho que los vectores que los representan (cada uno de los cuales define un eje) son ortogonales y – por ende – el coseno del ángulo que forman es igual a 0 (ya que α = 90º).

Sin embargo, el modelo vectorial es ampliamente utilizado ya que aporta visibles ventajas respecto del modelo booleano. Principalmente:

- El esquema de pesos mejora las prestaciones de la recuperación. Aquí se adquiere gran flexibilidad al poder incorporar diferentes esquemas de cálculo de los pesos.
- Se pueden considerar búsquedas aproximadas
- La medida de similitud proporciona un método de ranking de los resultados

- Mediante esta representación se puede medir la similitud entre diferentes objetos (documentos y consultas, documentos y documentos, oraciones y consultas, etc.)

Por otro lado, la simplificación realizada al considerar la independencia de los términos se puede ver como una desventaja. Además, asume que los documentos y las consultas son tratadas de igual manera y no existe una justificación que la soporte (tanto la longitud como el proceso de creación de ambos son completamente diferentes). No obstante estas observaciones, el modelo vectorial es superior o al menos equivalente al booleano.

4.2 – Medidas de semejanza

Hasta aquí, hemos planteado la semejanza entre consulta y documentos a partir del cálculo del coseno del ángulo que forman los vectores que los representan debido a que es la más utilizada. Sin embargo, esta no es la única medida. A continuación, se presentan otras medidas de semejanza utilizadas [Peña], aunque en menor medida:

Producto escalar: $\sum_{j=1}^{t} w_{dij} \bullet w_{qj}$

Coeficiente de Dice: $\dfrac{2 \bullet \sum_{j=1}^{t} w_{dij} \bullet w_{qj}}{\sum_{j=1}^{t} w_{dij}^2 + \sum_{j=1}^{t} w_{qj}^2}$

Coeficiente de Jaccard: $\dfrac{\sum_{j=1}^{t} w_{dij} \bullet w_{qj}}{\sum_{j=1}^{t} w_{dij}^2 + \sum_{j=1}^{t} w_{qj}^2 - \sum_{j=1}^{t} w_{dij} \bullet w_{qj}}$

A continuación, presentamos otro ejemplo donde se comparan documentos y consulta utilizando todas las medidas de similitud vistas. Suponga que se cuenta con un corpus C que consta de cuatro documentos (d_1, d_2, d_3 y d_4) y una consulta (q). En este caso solo hay dos términos posibles (A y B). Se tiene una matriz de asociación entre términos y documentos donde se han asignado los pesos correspondientes.

Documentos /Términos	**A**	**B**
d_1	1.0	1.0
d_2	0.5	1.0
d_3	1.0	0.8
d_4	0.7	0.7
d_5	1.0	0.0
q	1.0	1.0

	Coseno De α	**Producto Escalar**	**Coef. de Dice**	**Coef. de Jaccard**

$d(d_1,q)$	1.00	1º	2.00	1º	2.00	3º	1.00	4º
$d(d_2,q)$	0.95	4º	1.50	3º	2.40	2º	1.75	2º
$d(d_3,q)$	0.99	3º	1.80	2º	2.00	4º	1.84	1º
$d(d_4,q)$	1.00	2º	1.40	4º	2.86	1º	1.58	3º
$d(d_5,q)$	0.71	5º	1.00	5º	2.00	5º	0.50	5º

Aquí hay que tener en cuenta que – a igual valor del coeficiente de similitud – el ranking lo determinó el orden de evaluación. Nótese la diferencia en el ranking que se obtienen con las diferentes medidas. No obstante – en casos generales – la similitud por el coseno permite obtener el mejor rendimiento del sistema.

Una cuestión importante a destacar es que el cálculo de similitud por el coseno de los ángulos que forman ambos vectores es una medida que – a diferencia de la distancia por producto escalar – considera la normalización de los pesos de los vectores (esta operación corresponde al denominador de la expresión). La normalización se realiza a los efectos de evitar que los documentos más largos tengan mayores chances de ser recuperados y – de esta manera – se evita que la importancia de un término se base en la longitud del documento. Esto se debe a que es más probable que documentos más largos tengan más términos y más apariciones de éstos.

4.3 – Ejemplo completo de aplicación del modelo vectorial

En esta sección se presenta el desarrollo completo, paso a paso, del modelo vectorial, tanto de la representación de los documentos como de una consulta. Los pesos de los términos en los documentos se calcularon mediante la fórmula de TF*IDF.

El corpus consta de 5 documentos acerca de partes de automóviles:

Doc_1: {Puerta, Espejo, Caja}
Doc_2: {Puerta, Puerta, Filtro}
Doc_3: {Filtro, Espejo, Caja}
Doc_4: {Filtro, Rueda, Caja}
Doc_5: {Carter, Caja, Caja}

y la consulta:

Con_1: {Puerta, Filtro, Carter, Carter}

Se obtiene la matriz de frecuencias documentos– términos

	Rueda	**Carter**	**Espejo**	**Caja**	**Puerta**	**Filtro**
Doc_1			1	1	1	
Doc_2					2	1
Doc_3			1	1		1
Doc_4	1			1		1
Doc_5		1		2		

Luego, se obtiene la tabla de IDF:

Término	Frecuencia documento (df)	IDF (N = 5)
Rueda	1	2.32
Carter	1	2.32
Espejo	2	1.32
Caja	4	0.32
Puerta	2	1.32
Filtro	3	0.74

Finalmente, se calculan los pesos de los términos en cada documento por TF*IDF, combinando las tablas anteriores.

	Rueda	Carter	Espejo	Caja	Puerta	Filtro	Norma*
Doc_1			1.32	0.32	1.32		1.89
Doc_2					2.64	0.74	2.74
Doc_3			1.32	0.32		0.74	1.55
Doc_4	2.32			0.32		0.74	2.46
Doc_5		2.32		0.64			2.41

En esta tabla se agregó la columna norma, que corresponde a la longitud de los vectores y permite normalizar. Como se explicó anteriormente, se calcula como la raíz cuadrada de la suma del cuadrado de cada uno de los pesos. Ejemplo:

$$Norma(Doc_1) = \sqrt{(1.32^2) + (0.32^2) + (1.32^2)} = 1.89$$

En base a la consulta suministrada por el usuario, que incluye términos y frecuencias, se computan sus pesos:

	Rueda	Carter	Espejo	Caja	Puerta	Filtro	Norma
$Cons_1$ Frecuencias	0	2	0	0	1	1	-
$Cons_1$ Ponderada	0	4.64	0	0	1.32	0.74	4.88

A continuación, se calculan las semejanzas existentes entre la consulta y los cinco documentos.

$$SIM(Doc_1, Cons_1) = \frac{(0*0) + (0*4.64) + (1.32*0) + (0.32*0) + (1.32*1.32) + (0*0.74)}{(1.89*4.88)} = 0.18$$

$$SIM(Doc_2, Cons_1) = \frac{(0*0) + (0*4.64) + (0*0) + (0*0) + (2.64*1.32) + (0.74*0.74)}{(2.74*4.88)} = 0.30$$

$$SIM(Doc_3, Cons_1) = \frac{(0*0) + (0*4.64) + (1.32*0) + (0.32*0) + (0*1.32) + (0.74*0.74)}{(1.55*4.88)} = 0.07$$

$$SIM(Doc_4, Cons_1) = \frac{(2.32 * 0) + (0 * 4.64) + (0 * 0) + (0.32 * 0) + (0 * 1.32) + (0.74 * 0.74)}{(2.46 * 4.88)} = 0.04$$

$$SIM(Doc_5, Cons_1) = \frac{(0 * 0) + (2.32 * 4.64) + (0 * 0) + (0.64 * 0) + (0 * 1.32) + (0 * 0.74)}{(2.41 * 4.88)} = 0.91$$

Si se rankean las respuestas, por similitud, a la consulta cons1 se tiene el siguiente orden:

Similitud (d_j, q)	0,91	0,30	0,19	0,07	0,05
Documento	Doc_5	Doc_2	Doc_1	Doc_3	Doc_4
Orden de Relevancia	1	2	3	4	5

Del análisis de la tabla anterior surge que el documento 5 (Doc_5) es el mas similar a la consulta 1.

4.4 – Actividades

1) Modelo booleano. El siguiente es un extracto de un índice posicional que representa el contenido de un corpus de documentos bajo el formato:

término: doc1:posición1,pos2…; doc2:pos1,pos2, …; etc.

Donde posición hace referencia al número de caracter o byte en el cual empieza el término t para cada documento. Se sabe que un término t puede estar n veces en un documento d.

Corpus

a:	2:36,174,252,651;	4:12,22,102,432;	7:17;
gana:	2:67,124,393,1001;	4:11,41,101,421,431;	7:16,36,736;
inéditas:	2:3,37,76,444,851;	4:10,20,110,470,500;	7:5,15,25,195;
felicitaron:	2:57,94,333;	4:15,35,155;	7:20,320;
felicitas:	2:87,704,722,901;	4:13,43,113,433;	7:18,328,528;
fotos:	2:1,17,74,222;	4:8,78,108,458;	7:3,13,23,193;
la:	2:47,86,234,999;	4:14,24,774,944;	7:199,319,599,709;
rusas:	2:2,66,194,321,702;	4:9,69,149,429,569;	7:4,14,404;

1. Cuál/es documentos satisfacen la consulta ***"fotos rusas inéditas"***

2. Cuál/es documentos satisfacen la consulta ***"fotos rusas inéditas" AND "a felicitas la felicitaron"***

2) Un índice invertido posee alrededor de medio millón de documentos, el siguiente es un extracto de él:

Término	Frecuencia
E	213332
K	87029
M	107933
S	271678
TA	46673
TR	316832

Recomiende el orden de procesamiento para las siguientes consultas:

1. (TA OR TR) AND (M OR S) AND (K OR E)

2. TA AND (NOT M) AND (NOT TR)

3) ¿Cuál es la principal crítica que recibe el modelo booleano en lo referente a presentación de resultados al usuario?

4) Dados tres documentos con dos términos cada uno (D1 = (1, 3), D2 = (100, 300) y D3 = (3, 1)). calcule la semejanza entre a) d1 y d2 y b) entre d1 y d3. Use la métricas de

coeficiente del coseno y distancia de Euclides. Reflexione acerca de los resultados obtenidos.

5) Dado el siguiente corpus (en formato matriz documento-término)

	Guerra	Paz	Bush	Fidel	embargo	químico	misil	ONU	negocia
Doc1	2	3	4	2	0	0	2	4	1
Doc2	6	2	0	2	7	1	0	0	4
Doc3	0	1	8	2	0	0	0	2	1
Doc4	3	0	5	1	1	1	0	0	0
Doc5	2	1	7	7	2	9	2	0	1
Doc6	9	2	0	1	8	2	0	1	0
Doc7	8	2	0	2	0	1	4	2	0
Doc8	0	5	1	0	0	5	1	0	1
Doc9	0	5	1	9	2	0	0	5	1

Utilizando el modelo vectorial (con esquema de pesos TF-IDF) resuelva las siguientes consultas y muestre, en cada caso, el ranking de semejanza

Consulta 1= "Fidel, Bush, embargo"
Consulta 2 = "químico, misil, ONU"
Consulta 3 = "embargo, negocia, misil, Fidel"

6) Haga un programa que lea una matriz que representa a un corpus de documentos (como la existente en el ejercicio anterior), calcule el peso de los términos, reciba consultas de usuario y muestre el conjunto de documentos solución (ranking) para cada una.

7) Explique cual es la principal desventaja del modelo de recuperación vectorial

Capítulo 5

Estructuras de datos en RI

En este capítulo se presentan las estructuras de datos básicas para la implementación de sistemas de recuperación de información. A partir de los conceptos y las técnicas planteadas en el capítulo 3 sobre preprocesamiento de texto, resulta necesario contar con estructuras de datos eficientes que soporten las estrategias de búsquedas, de acuerdo al modelo de RI adoptado.

Como hemos mencionado, el SRI parte de un conjunto de documentos o colección, los cuales tiene que procesar – de alguna forma – para responder a consultas de los usuarios. De la manera más básica, se podría tomar la consulta del usuario y recorrer toda la colección, documento por documento, calculando la relevancia de cada uno. A esta técnica se la conoce como búsqueda secuencial o búsqueda por texto libre. Sin embargo, resulta ineficiente y solo es válida para colecciones textuales pequeñas o – bien – en aquellas que son de contenido dinámico o en casos en que no se dispone de recursos para gestionar el espacio de almacenamiento extra que requieren estructuras auxiliares.

Hay que tener en cuenta que si el sistema realiza operaciones de preprocesamiento como eliminación de palabras vacías y stemming, en la forma más ineficiente, estas tareas se repetirían para cada consulta. Entonces, se podría tener un conjunto de archivos con la representación lógica de los documentos de la colección, por ejemplo una tipo “bolsa de palabras” ya preprocesada por cada documento. Esto permite que las tareas previas se realicen solo una vez, pero el costo computacional de recorrer toda la colección sigue siendo alto (O(n) donde n es la cantidad de documentos que componen el corpus).

Con el objetivo de lograr mayor eficiencia en la tarea de recuperación, se construyen estructuras de datos auxiliares que soportan la representación lógica de todos los documentos de la colección. De forma genérica, estas estructuras reciben el nombre de índices. Como soporte del sistema, los índices permiten el acceso directo a los documentos que contienen los términos de la consulta. Como estructuras de datos, soportan diferentes técnicas de optimización, como por ejemplo la compresión.

El contenido del índice está dado por el conjunto de términos que contienen todos los documentos de la colección, es decir, el vocabulario. Una forma de representación de la información que debe contener un índice es la matriz documento–término (Figura 1). Sobre las filas se representan los documentos d_j y las columnas corresponden a los términos t_i del vocabulario. La intersección (d_j, t_i) corresponde al peso o ponderación que se le asigna a t_i en d_j. En el caso más simple puede ser un 0 o un 1, denotando ausencia o presencia del término en el documento o bien un valor surgido de un criterio de ponderación como – por ejemplo – TF-IDF.

Una distinción importante surge respecto de los sistemas de recuperación de datos como – por ejemplo – los sistemas de bases de datos. Éstos crean índices con los valores existentes en cada tabla de su clave primaria y sus claves secundarias. Como se puede deducir, se está aprovechando la estructura para determinar sobre qué atributos se crean índices. Por lo tanto, con éstos se optimizan las búsquedas, permitiendo rápido acceso a los ítems requeridos.

	t_1	t_2	t_3	…	t_n
d_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$		$w_{1,n}$
d_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$		$w_{2,n}$
d_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$		$w_{3,n}$
d_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$		$w_{4,n}$
d_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$		$w_{5,n}$
…					
d_n	$w_{n,1}$	$w_{n,2}$	$w_{n,3}$		$w_{n,n}$

Figura 1 – Matriz de asociación documento–término

Sin embargo, en un sistema de RI hay un inconveniente adicional. No existe un conjunto a priori de claves por la cuales buscar y – como no se puede saber de antemano cuáles se utilizarán en una búsqueda – todos los términos de todos los documentos son potenciales claves de búsqueda. De esta circunstancia surge que un índice para un sistema de RI contenga todo el vocabulario.

Por supuesto, el vocabulario estará formado por los términos que surgieron de las decisiones que se tomaron para el preprocesamiento de los documentos (eliminación de palabras vacías, stemming). Pero también hay que tener en cuenta cómo se realiza el proceso de identificación de los términos indexables (*tokenización*), ya que esto puede derivar en uno o dos términos o en formas modificadas. Por ejemplo, puntuación interna (gen-10 ó gen10) y mayúsculas/minúsculas (la plata ó La Plata).

Otra cuestión importante a tener en cuenta para la creación de índices es que su contenido varía de acuerdo al modelo de RI que debe soportar. De manera simple, la presencia de pares término/documento alcanza para el modelo booleano. Sin embargo, otros modelos requieren información estadística de la colección como frecuencia del término (*tf*), frecuencia en documentos (*df*), frecuencia en la colección (*ctf*), longitud de los documentos (*doclen*) y otras. Además, es posible que se requiera almacenar información posicional (para búsquedas por cercanía de términos) de cada término dentro del documento.

5.1 – Archivos invertidos

Partiendo de la matriz documento–término (Figura 1) se construye un índice que almacena su representación. La estructura de índice básica es la denominada “archivo invertido”. En su forma más simple, es un conjunto de términos donde cada uno tiene asociada una lista de los identificadores de documentos donde cada término aparece. Es decir, cada entrada en el archivo invertido mapea un término con un conjunto de documentos que lo contienen. Esta organización es completamente opuesta a la representación de cada documento como una bolsa de palabras, donde un documento está definido por un conjunto de términos. Entonces, para la construcción del índice,

podemos ver la matriz documento–término en forma invertida como matriz término-documento (Figura 2). A partir de ésta, la construcción del archivo invertido surge de forma directa.

	d_1	d_2	d_3	…	d_n
t_1	$w_{1,1}$	$w_{1,2}$	$w_{1,3}$		$w_{1,n}$
t_2	$w_{2,1}$	$w_{2,2}$	$w_{2,3}$		$w_{2,n}$
t_3	$w_{3,1}$	$w_{3,2}$	$w_{3,3}$		$w_{3,n}$
t_4	$w_{4,1}$	$w_{4,2}$	$w_{4,3}$		$w_{4,n}$
t_5	$w_{5,1}$	$w_{5,2}$	$w_{5,3}$		$w_{5,n}$
…					
t_n	$w_{n,1}$	$w_{n,2}$	$w_{n,3}$		$w_{n,n}$

Figura 2 – Matriz de asociación término–documento

El archivo invertido que soporta tal representación consta de dos partes: la primera es un término y la segunda la lista de documentos donde éste aparece, denominada lista de posteo (*posting list*). Nótese que el conjunto de todos los términos corresponde al vocabulario de la colección y – por supuesto – no tiene repetidos. En la figura 3 se muestra un ejemplo de una colección (con sus términos indexables por documento) y el archivo invertido resultante de la indexación.

d1 = {Se hacen soldadura de puertas}
d2 = {Reparo puertas y ventanas}
d3 = {Vendo ventanas usadas}
d4 = {Soldadura de puertas y marcos}

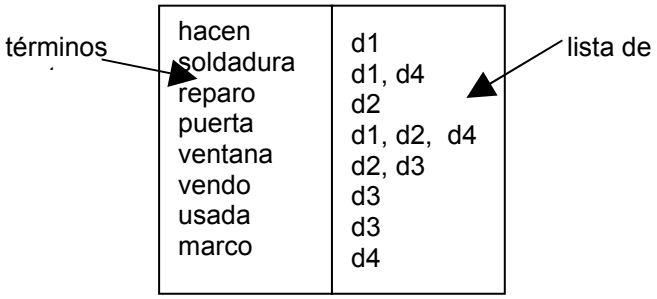


Figura 3 – Documentos y archivo invertido

La construcción de este índice es relativamente sencilla. En la figura 4 se muestra el algoritmo básico para esta tarea.

```
Por cada documento d en la colección
      Identificar los términos t en d
      Por cada término t en d
            Buscar t en el vocabulario
            Si existe t
                  Agregar d la lista de posteo de t
            Sino
                  Agregar t al vocabulario
                  Agregar d la lista de posteo de t

Grabar a disco el índice invertido
```

Figura 4 – Construcción de un índice invertido

Frakes [20] señala que la utilización de un archivo invertido aumenta la eficiencia en varios órdenes de magnitud al costo de tener que almacenar una estructura de datos cuyo tamaño varía entre el 10% y más del 100% del tamaño del texto.

La búsqueda en este archivo es – de la manera más simple – secuencial (O(n/2) siendo n el tamaño del vocabulario). Sin embargo, se puede implementar un índice invertido como un arreglo ordenado para permitir una búsqueda binaria (O(log n)). El costo mayor de mantenimiento de esta estructura está dado por las actualizaciones, aunque se considera que no son la tarea más común en un sistema de RI. Otra cuestión a tener en cuenta es que los índices pueden ser comprimidos a los efectos de reducir el espacio de almacenamiento requerido, sin embargo tiene un costo extra por la tarea de descompresión [22] [56].

Esta estructura es útil para resolver consultas utilizando el modelo booleano clásico, pero resulta inadecuada para modelos más sofisticados. Una alternativa es armar la lista de posteo con información acerca de la frecuencia del término t_i en cada documento d_j. En este caso, cada ítem de la *posting list* es un par ordenado (d_j, tf), donde d_j es el documento y tf es la frecuencia del término t_i en éste. En la figura 5 se muestra un ejemplo de archivo invertido con información de frecuencias.

t_1	(d_1, 2) (d_3, 4) (d_7, 8)
t_2	(d_2, 5) (d_7, 3)
t_3	(d_4, 1)
…	
t_{n-1}	(d_1, 1) (d_2, 2) (d_4, 5)
t_n	(d_{n-3}) (d_{n-1})

Figura 5 – Archivo invertido con información de frecuencias

En este ejemplo, el término t_1 aparece dos veces en el documento d_1, cuatro veces en d_3 y ocho veces en d_7.

También hay que tener en cuenta el costo asociado al mantenimiento del índice, tanto de construcción, actualización y almacenamiento. El tamaño de un índice puede – incluso – ser equivalente al de la colección misma. Para una colección de *n* palabras, según la ley de Heaps, el tamaño del vocabulario o lexicón resulta $V = O(n^{\beta})$, con $0.4 < \beta < 0.6$; mientras que las listas de posteo alcanzan O(n), si al menos tienen una ocurrencia por palabra en el texto.

5.1.1 – Archivo invertido posicional

Un archivo invertido posicional ó índice posicional es una estructura de datos que incluye las posiciones o desplazamientos donde cada término ocurre, con respecto al inicio del documento. Esta estructura almacena información de localización de los términos, lo que permite – entre otras – búsquedas por cercanía de términos y soporta búsquedas por frases.

La estructura de la lista de posteo es un poco más compleja ya que contiene los documentos donde aparece cada término junto con las posiciones en bytes donde se encuentran (Figura 6).

t_1	(d_1: 2, 10) (d_3: 4, 23, 54, 101) (d_7: 16, 54, 122)
t_2	(d_2, 50, 178) (d_7, 20, 62)
t_3	(d_4, 100)
…	
t_{n-1}	(d_1, 44) (d_2, 211) (d_4, 251)
t_n	(d_1, 32) (d_{n-1}, 101)

Figura 6 – Archivo invertido posicional

Como se puede apreciar, la frecuencia de un término en un documento corresponde a la cantidad de posiciones donde aparece. Este tipo de estructura de datos permite soportar consultas por proximidad, por ejemplo:

Q_1 = {red **adyacente** computadora}
Q_2 = {red **cerca-de** computadora}
Q_3 = {red **a-n-términos** computadora}

En Q_1, el operador **adyacente** requiere que las posiciones de los términos sean consecutivas con lo que se podría tener un archivo invertido con:

Red	(d1, 10)
Computadora	(d1, 11)

En las otras consultas Q_2 y Q_3, la distancia entre ambos términos puede ser mayor a uno. El operador **cerca-de** se define a una distancia fija (por ejemplo, 5 términos), mientras que con el operador **a-n-términos** se puede especificar cuál es la distancia máxima que se acepta para ambos términos.

Una variante posible es la de almacenar los números de oración o párrafos donde ocurren los términos, en vez de su posición en bytes. Esto requiere que primero se reconozca la estructura de oración (o párrafo) y luego los términos que contiene. Con esta información se pueden soportar consultas de tipo:

Q_4 = {red **misma-oración** computadora}

Donde se exige que ambos términos se encuentren al menos en una de las oraciones del documento.

Los archivos invertidos posicionales soportan consultas en las que deban satisfacer “restricciones de distancia” entre los términos. Si bien brindan mayor flexibilidad en las búsquedas, el costo asociado está dado por el espacio de almacenamiento extra que se requiere para el índice.

5.2 – Implementación sobre archivos invertidos

5.2.1 – Modelo booleano

Las consultas en el modelo booleano son relativamente sencillas. Para cada término q_i de una consulta Q se deben extraer – accediendo al índice – las listas de posteo correspondientes, compuestas de identificadores de documentos. Luego, de acuerdo a los operadores lógicos involucrados (AND, OR, NOT) se deben realizar las operaciones de conjunto necesarias. Si las listas de posteo se encuentran ordenadas de forma creciente por identificador de documento, se optimiza el proceso de comparación. Finalmente, los documentos del conjunto resultante forman la respuesta del sistema.

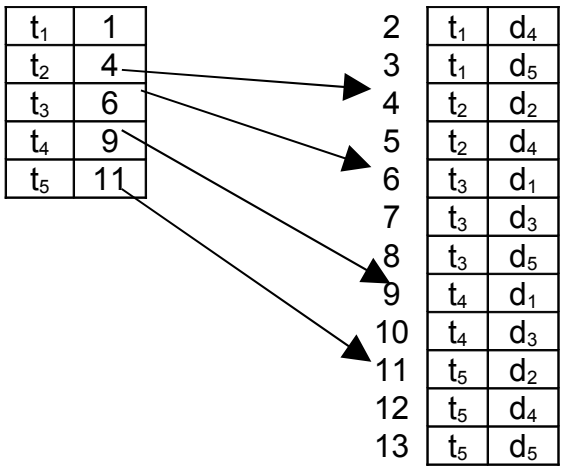
Como mencionamos anteriormente, en el modelo booleano básico la asociación entre un término t_i y un documento d_j ocurre cuando t_i se encuentra en d_j, sin importar su frecuencia. Por lo tanto, se construye la matriz de asociación binaria que tiene la siguiente forma:

	t_1	t_2	t_3	t_4	t_5
d_1	1	0	1	1	0
d_2	0	1	0	0	1
d_3	0	0	1	1	0
d_4	1	1	0	0	1
d_5	1	0	1	0	1

Esta estructura se puede almacenar fácilmente en un archivo invertido simple, como presentamos anteriormente. Sin embargo, se aprecia que no tiene sentido guardar las relaciones donde no ocurre un término en un documento (por ejemplo, [d1, t2]) por lo que tenemos una “matriz esparcida” y – de aquí – una posibilidad de almacenamiento más eficiente en un archivo de acceso directo. En éste, la clave primaria corresponde a la asociación término–documento existente ([d1, t2] = 1) en cada celda de la matriz.

T_n	d_n
t_1	d_1
t_1	d_4
t_1	d_5
t_2	d_2
t_2	d_4
t_3	d_1
t_3	d_3
t_3	d_5
t_4	d_1
t_4	d_3
t_5	d_2
t_5	d_4
t_5	d_5

Para lograr búsquedas eficientes, este archivo debe mantenerse ordenado a los efectos de recorrer solamente el bloque correspondiente al término buscado. Otra optimización consiste en agregar un nivel más de direccionamiento como en la organización multilista de archivos [30].



Otra estructura de datos que soporta al modelo booleano es un arreglo de bits [20], donde se cuenta con una forma compacta de almacenar la matriz de asociación término–documento. Aquí, cada documento es representado por una secuencia de bits, donde cada uno es asociado a un término. Si el i-ésimo término se encuentra en el documento el bit i-ésimo estará en 1, sino estará en 0. Para el ejemplo anterior, los vectores de bits resultantes son:

$V(d_1) = [10110]$
$V(d_2) = [01001]$
$V(d_3) = [00110]$
$V(d_4) = [11001]$
$V(d_5) = [10101]$

Luego, estos vectores se pueden almacenar como caracteres ASCII tomando cada 8 bits para representar un byte. A continuación, se presenta un ejemplo en lenguaje Perl de instrucciones para manejar vectores de bits, tanto su almacenamiento como operaciones lógicas entre éstos.

```
#Se parte de cadenas que representan documentos y términos (bit)

$d1 = "10110";
$d2 = "01001";

#Primero, se representan las cadenas en formato binario
$vector1 = pack("b*", $d1); # $vector1 es un escalar que contiene a
$vector2 = pack("b*", $d2); # d1 en formato binario

# Luego, se pueden realizar operaciones lógicas entre vectores de bits:
$and = $vector1 & $vector2;      # Se realiza un and (*)
$or  = $vector1 | $vector2;      # Se realiza un or (+)

# Para obtener los resultados, se mueven los escalares a un arreglo:
@res1 = split(//,unpack("b*", $and)); # Al arreglo res1 se transfiere el
@res2 = split(//,unpack("b*",  $or)); # vector de bits

foreach $elem (@res1) { print $elem }; print "\n";   # resultado del AND
foreach $elem (@res2) { print $elem }; print "\n\n"; # resultado del OR

# Con la funcion vec es posible setear un bit particular en 0 o 1
```

```
vec($vector1, 0, 1) = 0;  # se setea a 0 el primer bit (a la izquierda)
vec($vector2, 7, 1) = 1;  # se setea a 1 el ultimo bit de d2

# Se obtienen nuevamente los resultados:

@res1 = split(//,unpack("b*", $vector1));
@res2 = split(//,unpack("b*", $vector2));

foreach $elem (@res1) { print $elem }; print "\n";
foreach $elem (@res2) { print $elem }; print "\n\n";

# Con la funcion vec también es posible chequear el estado de un bit
particular:

for($i=0;$i<=7;$i++)
{
   $val1 = vec($vector1, $i, 1);       # se chequea el bit iesimo
   print $val1."\t"
}
print "\n";
```

Nótese la simpleza que se puede alcanzar con esta implementación ya que las operaciones lógicas son directas y – en este caso – pertenecen al lenguaje de programación. Este enfoque es eficiente cuando se opera sobre corpus de documentos de tamaño pequeño y se pueden almacenar todos los vectores de bits en memoria principal.

5.2.2 – Modelo vectorial

Como hemos visto, en el modelo vectorial se requiere de información de frecuencias de los términos y su aparición en los documentos. En este caso se cuenta con un índice que almacene la tf de cada t_i en d_j y – para cada término – su idf. Además, para normalizar se requerirá el máximo tf.

La estructura de datos mínima requerida es un archivo invertido con información de frecuencias como el presentado anteriormente el cual – además – contenga el idf de cada término. Sin embargo, se requiere un archivo adicional con la información del máximo tf para cada doc. Suponga el siguiente ejemplo, donde la matriz de asociación documento – término es la siguiente:

	t_1	t_2	t_3	t_4	t_5
d_1	3	2	1	0	5
d_2	0	1	0	2	1
d_3	1	2	4	0	1
d_4	0	0	2	2	3
d_5	3	2	1	0	1

max tf
5
2
4
3
3

idf	0.222	0.097	0.097	0.398	0.000

Luego, se construye un archivo invertido con la información de frecuencias y el idf de cada término:

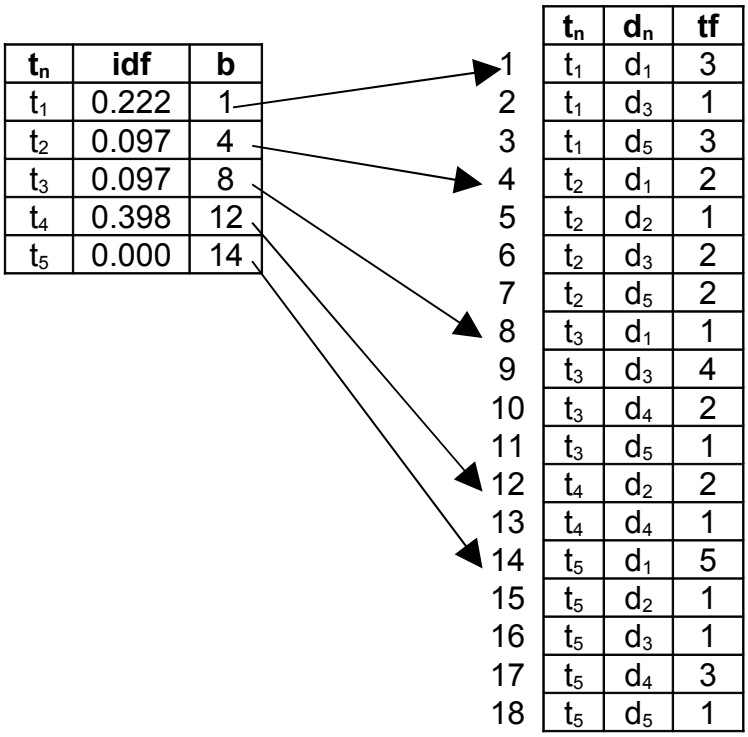
t_n	idf	posting list

t_1	0.222	(d_1:3); (d_3:1); (d_5:3)
t_2	0.097	(d_1:2); (d_2:1); (d_3:2); (d_5:2)
t_3	0.097	(d_1:1); (d_3:4); (d_4:2); (d_5:1)
t_4	0.398	(d_2:2); (d_4:1)
t_5	0.000	(d_1:5); (d_2:1); (d_3:1); (d_4:3) ; (d_5:1)

Y otro archivo con el máximo tf por término, el cual omitimos por su simplicidad (además, no es necesario si no se normalizan las frecuencias).

Con esta información almacenada, la búsqueda se realiza por cada término de la consulta sobre el archivo invertido, recuperando los pesos que permitan calcular la medida de similitud (sim(d_j, q)) y – de esta manera – realizar el ranking de los documentos. Opcionalmente, por cada término se recupera la máxima frecuencia de un término (max tf) para normalizar.

Al igual que lo planteado para el modelo booleano, es posible su almacenamiento en un archivo de acceso directo con la información de frecuencias, también con un nivel mayor de direccionamiento para aumentar la eficiencia.



Esta implementación es consistente con la propuesta de Grossman [23] donde planteó integrar datos estructurados y texto sobre el modelo relacional en un sistema de RI denominado SIRE. En este trabajo, Grossman definió un diseño relacional consistente de 6 tablas (relaciones) donde se representan los documentos y consultas:

- DOC (doc_id, doc_name, date, dateline): Contiene información sobre los documentos

- DOC_TERM (doc_id, term, tf): Mantiene la relación de aparición de un término dentro de un documento, junto con su frecuencia.

- DOC_TERM_PROX (doc_id, term, offset): Mantiene la relación de aparición de un término dentro de un documento, junto con su posición dentro de éste.

- IDF (term, idf): Almacena el valor de IDF de cada término en la colección.

- QUERY (term): Contiene los términos de cada consulta.

- STOP_TERM (term): Contiene una lista de las palabras vacías.

Nótese que las tablas DOC_TERM y DOC_TERM_PROX pueden ser vistas como los índices invertidos del sistema de recuperación, donde la segunda es equivalente a un índice invertido posicional.

El sistema soporta consultas basadas tanto en el modelo booleano como en otros modelos más avanzados que soportan ranking por relevancia. Por ejemplo, para el modelo booleano la expresión que permite obtener los documentos que contienen todos los términos de la consulta (AND) es la siguiente:

```
SELECT d.doc_id
FROM DOC_TERM d, QUERY q
WHERE d.term = q.term
GROUP BY d.doc_id
HAVING COUNT(DISTINCT(d.term)) = (SELECT COUNT(*) FROM QUERY)
```

De manera similar, si se desea construir una expresión con el operador lógico OR, se modifica la cláusula HAVING por una que permita especificar cuántos términos como mínimo (umbral) deben tener los documentos de la respuesta. Esto es:

```
HAVING COUNT(DISTINCT(d.term)) >= k
```

Donde k corresponde al valor del umbral.

Por otro lado, el sistema también soporta consultas por relevancia utilizando el modelo vectorial. Para ello, la tabla QUERY es extendida para incluir – además del término – su frecuencia (tf). Luego, la expresión en SQL es la siguiente:

```
SELECT d.doc_id, SUM(q.tf * i.idf * d.tf * i.idf)
FROM DOC_TERM d, QUERY q, IDF i
WHERE q.term = i.term
AND d.term = i.term
GROUP BY d.doc_id
ORDER BY 2 DESC
```

Además, se pueden modificar las expresiones SQL para rankear mediante otra medida de similitud o para implementar la normalización de las frecuencias [57]. En algunos casos, se requiere que se calculen datos extra de la colección como longitud promedio de los documentos, cantidad de documentos y tf promedio.

La evaluación de este sistema mostró que el enfoque adoptado utilizando el modelo relacional de bases de datos es una solución válida y portable que ofrece performance y escalabilidad para la implementación de SRI.

Actualmente, algunos sistemas de gestión de bases de datos relacionales (SGBD) cuentan con soporte para la indexación y recuperación de documentos de texto. Esto se conoce como “full text search” y es una característica que se ha incorporado – por ejemplo – a los SGBD MySQL y PostgreSQL. En el anexo 5 se presenta la implementación de la herramienta “Full Text Search” en MySQL.

5.3 – Actividades

1) Escriba un programa que tome un conjunto de documentos de un directorio, extraiga los términos y arme los índices que permitan soportar búsquedas mediante el modelo booleano. Utilice una lista de posteo sobre un archivo secuencial. (puede utilizar la librería Perl Tokenize). Luego, codifique un segundo programa que permita buscar por uno o dos términos.

2) Escriba un programa que tome un conjunto de documentos de un directorio, extraiga los términos y arme vectores de bits en memoria. Sobre éstos, implemente búsquedas booleanas. Escriba – luego – una función que almacene los vectores de bits a disco y otra que los recupere, para que la indexación sea persistente. Integre todo en un único programa.

3) Utilizando el programa anterior ejecute corridas con diferentes colecciones. Calcule los tamaños mínimos, máximos y promedio de las listas de posteo. ¿Qué utilidad tiene esta información?

4) Calcule la relación de overhead de los índices respecto de la colección. Calcule el overhead para cada documento. Luego, determine mínimos, máximos y promedio.

5) Agregue documentos a una colección (indexación incremental) y repita el ejercicio 3. Sus resultados: ¿Son consistentes con la ley de Heaps?.

6) La indexación incremental sobre archivos invertidos es una operación costosa. ¿Por qué? ¿Cómo se puede realizar eficientemente?

7) Modifique el programa del ejercicio 1 para armar un archivo invertido posicional a nivel de palabra. Luego, implemente consultas con operadores de proximidad.

8) Obtenga una copia del libro “Don Quijote de la Mancha” desde el Proyecto Gutenberg (http://www.gutenberg.org/etext/2000). Escriba un programa que extraiga los términos (no elimine las palabras vacías) y arme el índice invertido. Luego, calcule la distribución de frecuencias por palabra y evalúe – según Zipf y Luhn – los umbrales de corte y el conjunto de términos indexables. Con esta nueva información filtre el vocabulario y arme nuevamente el índice. ¿A qué tamaño y en qué proporción se redujo? Repita el ejercicio con una colección de textos científicos. ¿Qué diferencias encuentra? Justifique. Repita el ejercicio para textos en inglés.

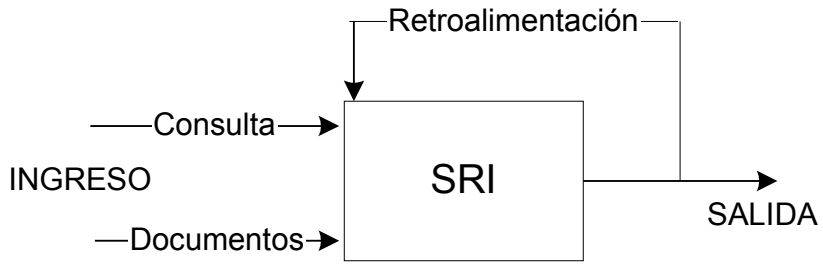
9) Modifique el programa del ejercicio 1 para armar un archivo invertido con información de frecuencias. Luego, implemente consultas utilizando el modelo vectorial utilizando tres fórmulas de ranking diferentes.

10) Ejecute una consulta en un motor de búsqueda web como – por ejemplo – Google. Recupere solo las 20 primeras páginas HTML del resultado. Escriba un programa que procese dichas páginas, elimine los tags HTML, extraiga el texto y arme un índice invertido con información de frecuencias. Utilice su programa del ejercicio 7 y ejecute la misma consulta y compare los rankings.

Capítulo 6

Retroalimentación de la consulta

La retroalimentación de la consulta [27] [7] [59] [29] es una operación ordenada por el usuario de un SRI, basada en la modificación de la consulta, que tiene por finalidad ajustar la formulación de la misma a los efectos de recuperar en un próximo intento nuevos documentos relevantes.



Una consulta puede ser modificada de dos formas a saber:

1) Por inclusión términos extraídos de un tesauro

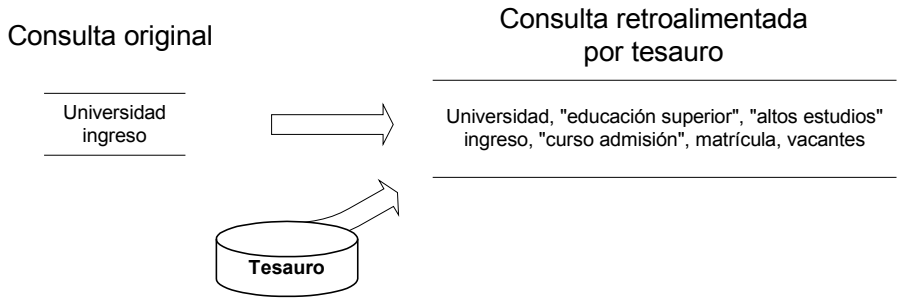
2) Por retroalimentación por relevancia: Se realiza generando un nuevo documento ficticio en base a la fusión de documentos reales que el usuario consideró relevantes.

6.1 – Expansión de la consulta con términos extraídos de un tesauro

En el proceso de armado de consultas ciertos factores pueden jugar en contra de las expectativas del usuario. Algunas situaciones de sinonimia y polisemia hacen que se pierda precisión y exhaustividad y – por ende – haya que reformular la consulta que expresa la necesidad de información del usuario.

Un elemento auxiliar en el proceso de expansión de la consulta es el tesauro. Éste es una herramienta documental, perteneciente a un área específica del conocimiento, que registra distintos tipos de relaciones entre términos. Tal herramienta contribuye a la reformulación de la consulta a partir de asesorar sobre nuevas formas de explicitar la consulta.

En un sistema de expansión de consultas por tesauro un usuario, ante una consulta donde se han obtenido resultados pobres, puede indicarle al SRI que incorpore n términos que tengan una alta asociación con los expresados en la consulta original. Tal incorporación se realiza de forma automática o manual, luego se envía la nueva consulta al SRI a los efectos de obtener nuevos objetos de información.



Un tesauro es un diccionario de términos controlados con relaciones explícitas entre ellos. En los SRI, su uso permite mejorar exhaustividad en la respuesta. Según la norma norma ISO 2788/TC, la cual define los principios para el desarrollo de tesauros monolingües, un tesauro provee:

1. Un vocabulario controlado y dinámico de términos que tienen entre sí relaciones semánticas (significados) y que se aplican a un campo particular del conocimiento.

2. Un instrumento de control de la terminología utilizada para trasladar el lenguaje escrito natural, utilizado en los documentos, a un lenguaje documental.

3. Un instrumento destinado al procesamiento de documentos de modo que garanticen las tareas destinadas a su provisión, almacenamiento y posterior recuperación.

En ciertas organizaciones el uso del tesauro se hace necesario debido a su característica de normalización del lenguaje específico al permitir la conversión del lenguaje técnico de un documento al propio lenguaje documental controlado. Todo usuario que accede a una base de datos en busca de información lo hará utilizando términos normalizados y así obtendrá mejor calidad de respuestas. El vocabulario controlado garantiza que:

1. Cada término descriptor posee un respaldo documental.

2. El lenguaje sea específico (preciso) para lograr una alta precisión.

3. El grupo descriptores que derivan de un documento sean representativos del mismo.

4. Por el uso de descriptores se elimine la ambigüedad.

Algunos ejemplos de tesauros disponibles en línea son:

- Tesauro de la UNESCO: http://www.ulcc.ac.uk/unesco/
- IRIS Keyword Thesaurus, Universidad de Illinois: http://carousel.lis.uiuc.edu/~iris/thesaurus.html
- AGROVOC, Tesauro de la FAO: http://www.fao.org/agrovoc/

Los elementos que conforman un tesauro son:

a) Descriptores: Son términos seleccionados para describir conceptos. Pueden estar formados por una o más palabras. El concepto de "identificador" se utiliza para representar nombres propios.

b) No Descriptores: Son sinónimos que se incluyen en el tesauro junto con la etiqueta "USE" que apunta al término base aceptado.

Ejemplo: Descriptor -> UNIVERSAL RESOURCE LOCATOR
No – descriptor -> URL

Las relaciones entre los descriptores pueden ser las siguientes:

1. Relaciones de Equivalencia: Se dan entre un término no-descriptor y un descriptor. Esta relación cubre dos tipos de términos: los sinónimos (USE) y los cuasi-sinónimos (UP). Los sinónimos son términos cuyo significado puede considerarse igual, por lo que son intercambiables. Los cuasi-sinónimos son palabras cuyo significado es considerado diferente en el uso común, pero son tratados como sinónimos en la indización. El indicador "used for" (UF) indica generalmente cual es la inversa de USE.

   Ejemplos:

   -URL
   USE UNIVERSAL RESOURCE LOCATOR

   -ASCENCION VERTICAL
   UP Ascensor
   UP Elevador
   UP Montacargas

   -UNIVERSAL RESOURCE LOCATOR
   UF URL

2. Relaciones Jerárquicas: Ocurren entre términos superiores y términos subordinados. Por ejemplo, en esta clase se utilizan las categorías "término amplio" (BT) y "término específico" (NT)

   Ejemplo:

   - UNIVERSAL RESOURCE IDENTIFIER
   BT Descripción formal
   NT Universal Resource Locator , Universal Resource Name

3. Relaciones Asociativas: Son aquellas relaciones que se dan entre términos en razón de un vínculo causa-efecto o tipo todo-parte. Un ejemplo de uso de la relación "término relacionado" (TR) es:

   - Selvicultura

TR Bosques

-Bosques
TR selvicultura

En la siguiente lista se proponen algunas categorías de relaciones usuales de los tesauros:

Denominación	*En inglés*	*En español*
Use / Use	USE	U
Use for / Use por	UF	UP
Broader term / Término amplio	BT	TA
Narrower term / Término específico	NT	TE
Related term / Término relacionado	RT	TR
Definitions / Definiciones	DEF	DEF
Note / Acotación	NOTE	NOTA

Ejemplo de entradas de un tesauro

STANDARD GENERAL MARKUP LANGUAGE (58)
DEF Standard for markup language applications
NOTE ISO 8879
UF SGML
BT Formal description

UNIVERSAL RESOURCE IDENTIFIER (65)
DEF Standard for construction of codes identifying resources or services on Internet
UF URI
BT Formal description
NT Universal resource locator , Universal resource name

UNIVERSAL RESOURCE LOCATOR (66)
DEF Code identifying a single copy of a document or another resource or service
UF URL
BT Universal resource identifier

URL
USE UNIVERSAL RESOURCE LOCATOR (66)

Desde el punto de vista de su presentación, un tesauro puede ser a) alfabético, donde los descriptores y los no descriptores se encuentran agrupados en una sola secuencia alfabética y b) sistemáticos, que presentan dos partes, una de ellas son las categorías y la otra un índice alfabético.

Tradicionalmente, el proceso de construcción de un tesauro puede dividirse en tres etapas a saber:

a) Construcción del vocabulario, Incluye la selección de términos (o frases) y su normalización.

b) Cálculo de las semejanzas entre los términos a los efectos de obtener asociaciones entre éstos.

c) Organización del vocabulario a partir de las asociaciones obtenidas en el segundo paso.

En lo referente a la construcción de tesauros de forma automatizada numerosos investigadores trabajaron sobre esto. Principalmente, emplearon técnicas que tienden a determinar la semejanza entre términos basándose en el estudio estadístico de co-ocurrencias [11] [17] [53]. En otros trabajos se realizaron análisis de conglomerados (clusters) para identificar grupos de términos relacionados por su semántica [46].

6.1.1 – Ejemplo de construcción automática de un tesauro

Una aproximación a la construcción automática de un tesauro puede realizarse a partir de identificar aquellos términos que ocurren en mismos documentos. Para ello, primero hay que construir una tabla binaria término-documento, donde el contenido de una celda sea 1 si el término ti está en el documento d_j y 0 por lo contrario:

	d_1	**d_2**	**d_3**	**d_4**	**d_5**
t_1	1	1	1		1
t_2		1	1		1
t_3	1		1		
t_4		1	1	1	
t_5	1	1	1	1	
t_6	1	1	1		

A continuación, hay que calcular el grado de asociación entre los términos, para lo cual es posible utilizar distintos coeficientes, los cuales miden el número de documentos en que ellos ocurren en comparación con el número de documentos que coocurren. Aquí se presentan tres modelos de cálculo posibles, los cuales ya se han utilizando antes en el cálculo de similitud entre consulta y documentos (Capítulo 3):

Coeficiente de Dice: $\dfrac{2 \bullet C_{ij}}{C_i + C_j}$

Coeficiente de Jaccard: $\dfrac{C_{ij}}{C_i + C_j - C_{ij}}$

Coeficiente del coseno: $\dfrac{C_{ij}}{\sqrt{C_i \times C_j}}$

Donde

C_i y C_j son el número de documentos donde T_i y T_j ocurren
C_{ij} es el numero de documentos donde coocurren los términos T_i y T_j

Los coeficientes retornan valores normalizados entre 0 y 1. Si dos términos ocurren exclusivamente en los mismos documentos su coeficiente es 1, pero si no co-ocurren su valor es 0. A continuación, se puede observar, a modo de ejemplo, el cálculo del grado de asociación entre el termino 1 y 2 del corpus hipotético:

Jaccard (t1, t2) = 3 / (4+3-3) = 0,75

Coseno (t1, t2) = 3 / (raiz_cuadrada(3*4)) = 0,86

Dice (t1, t2) = (3 * 2) / (4+3) = 0,85

Al coeficiente del coseno se lo conoce también como “cohesión” y fue propuesto por Salton y McGill. En general, este es el más utilizado para calcular el grado de asociación entre términos. Se presenta aquí la matriz término-término con todos los coeficientes calculados:

	t_1	t_2	t_3	t_4	t_5	t_6
t_1	1.00	**0,86**	0,70	0,57	0,75	**0,86**
t_2		1.00	0,40	0,66	0,57	0,66
t_3			1.00	0,40	0,70	0,81
t_4				1.00	**0,86**	0,66
t_5					1.00	**0,86**
t_6						1.00

En modo negrita se han destacado las asociaciones que se consideran relevantes. Para este ejemplo el umbral se fijo en 0.8 obteniéndose el siguiente conjunto A de asociaciones:

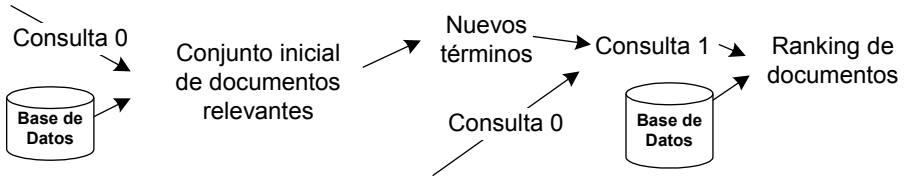
A = { (t1, t2), (t1, t6), (t4, t5), (t5, t6) }

Luego, utilizando alguna técnica de agrupamiento derivada del área de *clustering*, es posible obtener conjuntos de términos relacionados de más de dos elementos [**Peña**].

6.2 – Retroalimentación por relevancia

En la retroalimentación por relevancia el objetivo perseguido es agregar términos y generar una nueva consulta. Con ésta, en una nueva operación de recuperación, se espera obtener nuevos documentos relevantes.

Esto se logra a partir de que el usuario seleccione conjuntos de documentos relevantes y no relevantes, el sistema los analice y extraiga aquellos términos significativos. Con éstos se recalculan los pesos globales de todos los términos de la consulta de manera que ésta exprese más precisamente la necesidad de información del usuario. El proceso a realizar se ve en el siguiente gráfico:



6.2.1 – Algoritmo de Rocchio

Cuando se expande de forma automática una consulta realizada por un usuario ésta se realimenta utilizando aquellos documentos recuperados en la consulta inicial que el usuario ha señalado como relevantes. Rocchio [48] propuso una forma de cálculo de los pesos donde su característica principal es que permite ajustar la importancia que se desea dar a los términos de los documentos relevantes de la consulta original y también a los de los documentos que no se consideran relevantes.

Los pasos del algoritmo de Rocchio son los siguientes:

1) Identificar conjuntos de documentos relevantes y no relevantes a una consulta de usuario c.

2) Seleccionar nuevos términos del conjunto de documentos relevantes. En general se toman los que poseen mayor valor de TF*IDF.

3) Se adicionan los términos seleccionados en el paso 2 con los de la consulta original y se asígnan nuevos pesos a cada uno en base a una función de ponderación.

4) Se procesa la nueva consulta.

La función de ponderación de Rochio para recalcular el peso de un término en la nueva consulta es la siguiente:

$$Q'_i = (a \cdot Q_i) + (b \cdot \text{promedio}(R_i)) - (c \cdot \text{promedio}(S_i))$$

Donde:

Q_i	es el peso asignado al término i en la nueva consulta.
a, b, y c	son constantes de ponderación
R_i	es el peso del término i en el conjunto de documentos relevantes
S_i	es el peso del término y en el conjunto de documentos no relevantes

En la formula anterior existen diversas variantes para asignar valores a las constantes a, b y c, algunas son:

a=1, b=0,5, c=0,25
a=1, b=1/|R| , c=1/|S|
a=b=c=1
No utilizar el tercer término, haciendo c=0

Aquí se presenta un ejemplo completo de retroalimentación por el método propuesto por Rocchio. Suponga que se tiene el siguiente corpus de documentos

	t_1	t_2	t_3	t_4	t_5	t_6
d_1	1	0	1	0	0	0
d_2	3	0	2	1	0	0
d_3	1	2	3	0	1	0
d_4	0	0	0	2	1	2
d_5	1	1	1	4	2	1
d_6	1	1	0	0	3	2

t_n	IDF
t_1	0,26
t_2	1,00
t_3	0,58
t_4	1,00
t_5	0,58
t_6	1,00

Matriz TF*IDF	t_1	t_2	t_3	t_4	t_5	t_6
d_1	0,26	0,00	0,58	0,00	0,00	0,00
d_2	0,79	0,00	1,17	1,00	0,00	0,00
d_3	0,26	2,00	1,75	0,00	0,58	0,00
d_4	0,00	0,00	0,00	2,00	0,58	2,00
d_5	0,26	1,00	0,58	4,00	1,17	1,00
d_6	0,26	1,00	0,00	0,00	1,75	2,00

Consultas	t_1	t_2	t_3	t_4	t_5	t_6
Q_1	1,60	0,00	0,36	0,00	0,00	0,00
Q_2	0,00	0,00	0,00	0,60	0,00	0,60

Luego de procesadas las consultas Q_1 y Q_2 se obtienen los siguientes valores de semejanza:

Documento	**Q1**	**Q2**
d_1	0,60	0,00
d_2	0,59	0,41
d_3	0,23	0,00
d_4	0,00	0,98
d_5	0,09	0,79
d_6	0,09	0,50

Los rankings presentados son:

Posición	**Q1**	**Q2**
1	d1	d4
2	d2	d5
3	d3	d6
4	d6	d2
5	d5	d1
6	d4	d3

El usuario selecciona como relevantes los siguientes documentos:

	Ranking Q_1
d_1	REL
d_2	REL
d_3	NO REL
d_6	NO REL
d_5	NO REL
d_4	NO REL

	Ranking Q_2
d_4	NO REL
d_5	NO REL
d_6	REL
d_2	NO REL
d_1	NO REL
d_3	REL

Con esta información, se crea un nuevo vector $q_1´$, donde cada elemento esta formado por:

> $q_1´\_n$ = el valor original del termino n en la consulta original más la sumatoria del promedio de los pesos del termino n en docs relevantes menos la sumatoria del del promedio de los pesos del termino n en docs no relevantes.
>
> En nuestro ejemplo los pesos a, b y c = 1 (normalmente son a=1, b=0,5 y c=0,25)

Ahora se calcula el nuevo vector $q_1´$:

Relevantes	t_1	t_2	t_3	t_4	t_5	t_6
d_1	0,26	0,00	0,58	0,00	0,00	0,00
d_2	0,79	0,00	1,17	1,00	0,00	0,00
Promedio	0,53	0,00	0,88	0,50	0,00	0,00

No Relevantes	t_1	t_2	t_3	t_4	t_5	t_6
d_3	0,26	2,00	1,75	0,00	0,58	0,00
d_4	0,00	0,00	0,00	2,00	0,58	2,00
d_5	0,26	1,00	0,58	4,00	1,17	1,00
d_6	0,26	1,00	0,00	0,00	1,75	2,00
Promedio	0,20	1,00	0,58	1,50	1,02	1,25

Q_1	1,60	0,00	0,36	0,00	0,00	0,00

$Q_1´$	1,93	-1,00	0,65	-1,00	-1,02	-1,25

Con esta nueva consulta se calcula la semejanza de todos los documentos y se obtiene el siguiente ranking producto de la retroalimentación:

	Semejanzas
$SIM(d_1,q_1´)$	0,47
$SIM(d_2,q_1´)$	0,25
$SIM(d_3,q_1´)$	-0,12
$SIM(d_4,q_1´)$	-0,60
$SIM(d_5,q_1´)$	-0,50
$SIM(d_6,q_1´)$	-0,57

Ranking $Q_1´$
d_1
d_2
d_3
d_5
d_6
d_4

De la misma forma se procesa la consulta q2, resultando:

Relevantes	t_1	t_2	t_3	t_4	t_5	t_6
d_3	0,26	2,00	1,75	0,00	0,58	0,00
d_6	0,26	1,00	0,00	0,00	1,75	2,00
Promedio	0,26	1,50	0,88	0,00	1,17	1,00

No Relevantes	t_1	t_2	t_3	t_4	t_5	t_6
d_1	0,26	0,00	0,58	0,00	0,00	0,00
d_2	0,79	0,00	1,17	1,00	0,00	0,00
d_4	0,00	0,00	0,00	2,00	0,58	2,00
d_5	0,26	1,00	0,58	4,00	1,17	1,00
Promedio	0,33	0,25	0,58	1,75	0,44	0,75

Q_2	1,60	0,00	0,36	0,00	0,00	0,00

Q_2'	1,53	1,25	0,65	-1,75	0,73	0,25

	Semejanzas
SIM(d_1,q_2')	0,43
SIM(d_2,q_2')	0,04
SIM(d_3,q_2')	0,57
SIM(d_4,q_2')	-0,31
SIM(d_5,q_2')	-0,30
SIM(d_6,q_2')	0,42

Ranking Q_2'
d1
d3
d2
d6
d5
d4

Como se vio, el método de Rocchio propone la generación de un nuevo documento ficticio en base al refinamiento de la consulta. En teoría, en cada iteración, se refleja de manera más precisa la necesidad de información del usuario. Aquí también es necesario reconocer que es importante la selección de documentos relevantes por parte del usuario ya que – ante una mala elección – se puede desviar negativamente la siguiente consulta.

6.3 – Actividades

1) ¿Por qué el uso de un tesauro en la operación de consulta a un SRI mejora la exhaustividad?

2) Use un sistema de recuperación de información que soporte la retroalimentación por relevancia. Experimente con cinco consultas distintas. Evalúe los resultados obtenidos en cada caso.

3) Tome un corpus de 20 noticias cortas relacionadas entre si e investigue las relaciones entre términos. ¿Es posible detectar y extraer términos relacionados?

4) Desarrolle un ejemplo de retroalimentación por relevancia (método Rocchio). Utilice el siguiente corpus

	T1	T2	T3	T4	T5	T6	T7	T8
D1	3	0	1	0	0	0	0	0
D2	2	0	0	4	0	0	4	0
D3	0	1	0	1	4	0	1	4
D4	1	1	0	3	2	3	3	2
D5	1	2	0	0	3	0	0	0
D6	2	0	3	3	0	4	0	1
D7	0	0	0	0	1	1	4	0
D8	0	4	0	0	0	3	2	3
D9	0	1	4	4	0	1	0	0
D10	4	3	2	1	4	0	1	0
Q1	0.5	1	0	0.8	0	1	1.5	0
Q2	0	0	1,5	0	1	0,5	0,3	1

5) ¿Por qué se dice que Rocchio utiliza un “documento virtual o ficticio” como elemento clave de su técnica?

Recursos de interés relacionados con la recuperación de información

Software:

Karpanta

Motor de recuperación de información experimental desarrollado por el Grupo Reina (Universidad de Salamanca).
http://reina.usal.es/materiales/karpanta/index.htm

Okapi

Software de indexación de colecciones de documentos.
http://www.soi.city.ac.uk/~andym/OKAPI-PACK/

Bow

Es una herramienta, diseñada por Andrew McCallum's, destinada a ayudar al estudio de modelos de lenguaje, recuperación de información, clasificación y agrupamiento de documentos. Se distribuye en código fuente bajo licencia GNU.
http://www.cs.cmu.edu/~mccallum/bow

TnT

Etiquetador léxico que corre sobre plataformas Solaris y Linux.
http://www.coli.uni-sb.de/~thorsten/tnt/

Editor XML

jEdit: es un editor de archivos en formato XML de licencia libre. Oisee plugins para soportar extensiones del lenguaje.
http://www.jedit.org/

CLUTO

Paquete orientado a tareas de análisis de conglomerados (clustering). Corre en plataformas Linux, Sun y Windows 2000.
http://www-users.cs.umn.edu/~karypis/cluto/download.html

NLTK

Clasificador que incorpora diversos métodos (Naive Bayes, MaxEnt, etc).
http://nltk.sourceforge.net

Orange

Software de minería de datos desarrollado en C++. Incluye técnicas de SVM, análisis de conglomerados, regresión logística, etc.
http://magix.fri.uni-lj.si/orange/.

WebSPHINX

Web crawler de uso moderado.
http://www-2.cs.cmu.edu/~rcm/websphinx/

Balie

Herramienta destinada a la extracción de información.
http://balie.sourceforge.net/

SWISH-E

Motor de consulta
http://www.searchtools.com/tools/swish-e.html

XML Query Engine

Motor de consulta que indexa documentos en formato XML
http://www.searchtools.com/tools/xqengine.html

ht://Dig

Motor de consulta de uso popular.
http://www.searchtools.com/tools/htdig.html

Snowball

Herramienta para desarrollo de stemmers.
http://snowball.tartarus.org/

Weka

Software de soporte a la minería de datos. Desarrollado en java.
http://www.cs.waikato.ac.nz/~ml/weka

MG

Managing Gigabytes, software de recuperación de información.
http://www.math.utah.edu/pub/mg/

Lucene

Motor de consulta freeware, codificado en Java.
http://jakarta.apache.org/lucene/docs/index.html

SMART

Sistema clásico de recuperación de información.
ftp://ftp.cs.cornell.edu/pub/smart/

Conferencias

CLEF Cross-Language Evaluation Forum
http://www.clef-campaign.org/

DUC Document Understanding Conferences.
http://www-nlpir.nist.gov/projects/duc/

INEX Initiative for the Evaluation of XML retrieval.
http://www.is.informatik.uni-duisburg.de/projects/inex03/

TREC Text Retrieval Conference.
http://trec.nist.gov/

ISMIR Conferencia sobre técnicas de IR aplicadas a la música.
http://ismir2002.ircam.fr/

Juegos de datos (datasets)

Chamaleon Orientado a pruebas de análisis de conglomerados
http://www-users.cs.umn.edu/~karypis/cluto/files/
chameleon-data.tar.gz

SMART Colección de distintos juegos de datos destinados a probar el sistema de recuperación de información SMART.
ftp://ftp.cs.cornell.edu/pub/smart/

Medline
Corpus de prueba de sistemas de recuperación de información ad-hocs. Es orientado a la medicina.
http://www.nlm.nih.gov/databases/databases_medline.html

Reuters Reuters 21578, corpus destinado a evaluar sistemas de clasificación.
http://www.research.att.com/~lewis/reuters21578.html

4-U Conjunto de 8.282 páginas web obtenidas de cuatro universidades y clasificadas por tipo de páqina.
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

Oshumed Colección de documentos médicos cuyo objetivo es evaluar SRIs.
http://trec.nist.gov/data.html

20 NW 20 newsgroups datasets. Colección de 20.000 noticias destinada a evaluar técnicas de clasificación de documentos.
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/news20.html

Glasgow Juegos de prueba para evaluar SRIs alojados en la Universidad de Gasglow
http://ir.dcs.gla.ac.uk/resources/test_collections/

Grupos de investigación destacados

Grupo Reina
Grupo de investigación en REcuperación de la INformación Automatizada.
Universidad de Salamanca, España.
http://reina.usal.es

Glasgow Information Retrieval Group
Universidad de Glasgow, Inglaterra
http://ir.dcs.gla.ac.uk/

Intelligent Internet Research
IIT Bombay, India
http://www.cse.iitb.ac.in/laiir/

Multimedia Knowledge Management
Imperial College London, Inglaterra
http://km.doc.ic.ac.uk/

Grupo de investigación de técnicas de gerenciamiento de la información
Universidad de Padova, Italia
http://www.dei.unipd.it/~ims

Information Access Lab
University of Strathclyde, Inglaterra
http://www.ilab.cis.strath.ac.uk/

Center for Intelligent Information Retrieval
University of Massachusetts, USA
http://ciir.cs.umass.edu/

Language Technologies Institute
School of Computer Science - Carnegie Mellon University, USA
http://www.lti.cs.cmu.edu/

Referencias

[1] Adamson G. y Boreham. J. *"The use of an association measure based on character structure to identify semantically related pairs of words and document titles"*. In Information Processing & Management. vol. 10(7/8), pags. 253-260, 1974.

[2] Baeza-Yates, R. y Ribeiro-Neto, B. "*Modern Information Retrieval*". ACM Press. Addison Wesley. 1999.

[3] Blair, D.C. "*The Data-Document Distinction in Information Retrieval*" Communications of the ACM, v. 27:4.1984.

[4] Blair, D.C. "*The Data-Document Distinction Revisited*". University of Michigan Working Paper. 1999.

[5] Bordignon, F.R.A.; Panessi, W. *"Procesamiento de variantes morfológicas en búsquedas de textos en castellano"*. Revista Interamericana de Bibliotecología, 24 (1) pags. 69-88. 2001

[6] Brewington, B. E. y Cybenko Thayer, G. "*How Dynamic is the Web?*" En: Proceedings of the Ninth International World Wide Web Conference. 2000.

[7] Buckley, C.; Salton, G. y Allan, J. *"The effect of adding relevance information in a relevance feedback environment".* Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, pags. 292-300. 1994.

[8] Burkowski, F. "*Retrieval activities in a database consisting of heterogeneous collections of structured texts*". In Belkin, N., Ingwersen, P., Pejtersen, A. M., and Fox, E., editors, Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, págs. 112–125, New York. ACM Press. 1992.

[9] Carlson, C. "*Information overload, retrieval strategies and Internet user empowerment*".In Haddon, Leslie, Eds. Proceedings The Good, the Bad and the Irrelevant (COST 269) 1(1), pp. 169-173, Helsinki (Finland). 2003.

[10] Cavnar, W.B. *"N-Gram-based Text Filtering for TREC-2"*. 2nd Text Retrieval Conference (TREC-2), NIST Special Publication 500-215, National Institute of Standards of Technology, Gaitherbug, Maryland. 1994.

[11] Chen, H. y Lynch, K.J. *"Automatic construction of networks of concepts characterizing document databases"*. IEEE Transactions on Systems, Man and Cybernetics, 22(5), pags. 885-902. 1992.

[12] Church, K. W. y Hanks, P. *"Word association norms, mutual information, and lexicography".* Computational Linguistics 16(1), pags. 22-29. 1990.

[13] Cleverdon, C.W., Mills, J. Y Keen, M. "*Factors Determining the Performance of Indexing Systems*". ASLIB Cranfield Project. Vol. 1, Design, Vol2, Test Results. 1966.

[14] Cleverdon, C.W. "*On the inverse relationship of recall and precision*". Journal of Documentation, vol. 28, págs. 195-201. 1972.

[15] Cormack, G.V., Palmer, C.R., Clarke, L. A. "*Efficient Construction of Large Test Collections*". Proceedings of Melbourne SIGIR 1998. Conference on Research and Development in Information Retrieval. ACM Press. Págs. **???**. 1998.

[16] Croft, W.B. *"Approaches to intelligent information retrieval."* Information Proccesing & Management, 23, 4, pp. 249-254. 1987.

[17] Crouch, C.J. *"An approach to the automatic construction of global thesauri"*. Information Processing and Management, 26(5), pags. 629-640, 1990.

[18] Egnor, D. y Lord, R. "*Structured information retrieval using XML*". En: Proceedings of the ACM SIGIR 2000 Workshop on XML and Information Retrieval, Julio 2000.

[19] Figuerola, C. G.; Zazo, A. F.; Rodriguez Vázquez de Aldana, E. Y Alonso Berrocal, J.L. *"La Recuperación de Información en español y la normalización de términos".* Revista Iberoamericana de Inteligencia Artificial, v.22, n.8. pags. 135-145. 2003.

[20] Frakes, W.B. y Baeza-Yates, R. *"Information Retrieval Data Structures and Algorithms"*. Prentice Hall. 1992.

[21] Francis, W. y Kucera, H. *"Frequency analysis of english usage".* Lexicon and Grammar. Houghton Mifflin. 1982.

[22] Frieder, O.; Grossman, D. A.; Chowdhury, A. y Frieder, G. "*Efficiency considerations for scalable information retrieval servers*". Journal of Digital information, 1(5), 2000.

[23] D.A. Grossman, O. Frieder, D.O. Holmes, and D.C. Roberts. "*Integrating Structured Data and Text: A Relational Approach*". Journal of the American Society of Information Science, 48(2):122–132. 1997.

[24] Grossman, D. y Frieder, O. "*Information Retrieval. Algorithms and Heuristics*". Kluwer Academic Publishers. 1998.

[25] Korfhage, R. R. "*Information Storage and Retrieval*". New York. Wiley Computer Publishing. 1997.

[26] Harman, D. *"How efective is sufxing?"* Journal of the American Society for Information Science, 42(1). Pags. 7-15, 1991.

[27] Harman, D. *"Relevance Feedback and Others Query Modification Techniques"*, en Information Retrieval: Data structures and algorithms, Prentice-Hall, New Jersey, 1992.

[28] Hull, D. *"Stemming algorithms – a case study for detailed evaluation"*. Journal of the American Society for Information Science, 47(1). 1996.

[29] Jing, Y. y Croft, W. B. "*An association thesaurus for information retrieval".* In Proceedings of RIAO-94, 4th International Conference "Recherche d'Information Assistee par Ordinateur", pags 146–160, New York, US, 1994.

[30] Katzopoulos, M. y Kollias J.G. "*On the Optimal Selection of Multilist Database Structures*". En: IEEE Transactions of Software Engineering, v.10, n.6, pp681-687. 1984.

[31] Kraaij, W. y Pohlmann, R. *"Porter's stemming algorithm for dutch"*. In L. G. M. Noordman and W. A. M. de Vroomen, editors, Informatiewetenschap, pags. 167-180, Tilburg. 1994.

[32] Lancaster, F.W. "*Information retrieval systems: characteristics, testing and evaluation*". New York: John Wiley & Sons. 1979.

[33] Lawrence, S. y Giles, L. "*Accessibility and Distribution of Information on the Web*". Nature, vol.400, n.6740, pags.107-109. 1999.

[34] Luhn, H.P., *"The automatic creation of literature abstracts"*, IBM Journal of Research and Development, 2, 1pags. 59-165. 1958.

[35] Maes, P. "*Agents that Reduce Work and Information Overload*". Communications of the ACM, Vol. 37, Nro. 7, págs. 30-40. 1994.

[36] Martinez Mendez, F.J. y Rodriguez Muñoz, J.V. "*Reflexiones sobre la Evaluación de los Sistemas de Recuperación de Información: Necesidad, Utilidad y Viabilidad*". Anales de Documentación, Nro. 7, págs. 153-170. 2004.

[37] Navarro, G. y Baeza-Yates, R. "*A language for queries on structure and contents of textual databases*". Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, págs. 93-101, New York. ACM Press. 1995.

[38] Ogilvie, P. y Callan, J. "*Language Models and Structured Document Retrieval*". En: Proceedings of the first INEX workshop, 2003.

[39] Peña, R., Baeza-Yates, R., Rodriguez, J.V. *"Gestión Digital de la Información"*. Alfaomega Grupo Editor. 2003.

[40] Peterson, J.L. *"Computer Programs for Detecting and Correcting Spelling Errors"*. Comm ACM 23, 676. 1980.

[41] Pollit, A.S. "*Information storage and retrieval systems: origin, development and applications*". Toronto: John Wiley & Sons, pp. 164-65. 1989.

[42] Popovic, M. y Willett, P. *"The effectiveness of stemming for natural-language access to Slovene textual data"*. Journal of the American Society for Information Science, 43(5), pags. 384-390, 1992.

[43] Porter, M. F. *"An algorithm for suffix stripping"*. Program, 14(3), pags. 130-137. 1980.

[44] Radecki, T. "*Fuzzy Set Theoretical Approach to Document Retrieval*" Information Processing and Management, vol 15. pp 247-259. 1979.

[45] Raghavan, S. Y Garcia-Molina, H. "*Integrating diverse information management systems: A brief survey*". IEEE Data Engineering Bulletin, 24(4):44-52, 2001.

[46] Rasmussen, E. *"Clustering algorithms".* En: Information Retrieval: Data Structures and Algorithms, W. B. Frakes and R. Baeza-Yates, Editors, Prentice Hall, Englewood Cliffs, NJ, 1992.

[47] Robertson, S.E y Spark-Jones, K. "*Relevance Weighting of Search terms*". Journal of Documentation. 33, p.126-148. 1976.

[48] Rocchio, J. J. *"Relevance feedback in information retrieval"*. Salton, G. (ed.). The SMART retrieval system. Englewood Hills (NJ): Prentice-Hall, pags. 313-323. 1971.

[49] Salton, G. (editor). "*The SMART Retrieval System – Experiments in Automatic Document Processing*". Prentice Hall In. Englewood Cliffs, NJ. 1971.

[50] Salton, G. Y Mc Gill, M.J. "*Introducttion to Modern Information Retrieval*". New York. Mc Graw-Hill Computer Series. 1983.

[51] Salton, G.; Fox, E.A. y Wu, H. "*Extended Boolean information retrieval*". Communications of the ACM, 26(11):1022-1036. Noviembre, 1983.

[52] Salton, G. "*On the relationship between theoretical retrieval models*", Informetrics 87/88,. Diepenbeeck (Bélgica), pp. 263-270. 1987.

[53] Salton, G. "*Automatic Text Processing*". Addison Wesley, Reading, Mass. 1989.

[54] Sanderson, M. Y Joho, H. "*Forming Test Collections with No System Pooling*". Proceedings of Sheffield SIGIR 2004. Conference on Research and Development in Information Retrieval. ACM Press. Págs. 33-40. 2004.

[55] Schmitt, J.C. *"Trigram-based method of language identification"*. U.S. Patent 5,062,143. 1990.

[56] Scholer, F.; Williams, H. E.; Yiannis, J. y Zobel, J. "*Compression of inverted indexes for fast query evaluation*". En: Proceedings of the 25th Annual International ACM SIGIR conference on Research and development in information retrieval, pp 222--229, 2002.

[57] Singhal, A.; Buckley, C. y Mitra, M. "*Pivoted Document Length Normalization*". En: Proceedings of the 19th Annual International ACM SIGIR conference on Research and development in information retrieval. 1996.

[58] Smadja, F. *"Retrieving Collocations from Text: Xtract"*. Computational Linguistic, 19(1), pags. 143-177. 1993.

[59] Smeaton, A. y van Rijsbergen, C. *"The retrieval effects of query expansion on a feedback document retrieval system"*. The Computer Journal, 26(3), pags. 239–246, 1983.

[60] Mounir, S., Goharian, N., Mahoney, M., Salem, A., and Frieder, O. *"Fusion of Information Retrieval Engines (FIRE)*", International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), Las Vegas. 1998.

[61] van Rijsbergen, C.J. "*Information Retrieval*". Department of Computing Science. University of Glasgow. 1979.

[62] van Slype, G. "*Los lenguajes de indización. Concepción, construcción y utilización en los sistemas documentales".* Editorial Pirámide, Madrid. 1991.

[63] Voorhees, E.M. "*Variations in relevance judgments and the measurement of retrieval effectiveness*". Information Processing and Management, 36 (5), págs. 697-716. 2000.

[64] Waller, W. G. y Kraft, D. H. "*A mathematical model for a weighted Boolean retrieval system*". Information Processing and Management, Vol 15, No. 5, pp. 235-245. 1979.

[65] Yannakoudakis, E.J., Goyal, P. y Huggill, J.A. "*The Generation and Use of Text Fragments for Data Compression*". Information Processing and Management Volumen 18(1), 15-21. 1982.

[66] Zipf, G. K. *"Human Behaviour and the Principle of Least Effort"* Reading, MA: Addison- Wesley Publishing Co. 1949.

Anexos

Anexo 1 – Lista de palabras vacías de la lengua inglesa extraidas del sistema SMART

a a's able about above according accordingly across actually after afterwards again against ain't all allow allows almost alone along already also although always am among amongst an and another any anybody anyhow anyone anything anyway anyways anywhere apart appear appreciate appropriate are aren't around as aside ask asking associated at available away awfully b be became because become becomes becoming been before beforehand behind being believe below beside besides best better between beyond both brief but by c c'mon c's came can can't cannot cant cause causes certain certainly changes clearly co com come comes concerning consequently consider considering contain containing contains corresponding could couldn't course currently d definitely described despite did didn't different do does doesn't doing don't done down downwards during e each edu eg eight either else elsewhere enough entirely especially et etc even ever every everybody everyone everything everywhere ex exactly example except f far few fifth first five followed following follows for former formerly forth four from further furthermore g get gets getting given gives go goes going gone got gotten greetings h had hadn't happens hardly has hasn't have haven't having he he's hello help hence her here here's hereafter hereby herein hereupon hers herself hi himhimself his hither hopefully how howbeit however i i'd i'll i'm i've ie if ignored immediate in inasmuch inc indeed indicate indicated indicates inner insofar instead into inward is isn't it it'd it'll it's its itself j just k keep keeps kept know knows known l last lately later latter latterly least less lest let let's like liked likely little look looking looks ltd m mainly many may maybe me mean meanwhile merely might more moreover most mostly much must my myself n name namely nd near nearly necessary need needs neither never nevertheless new next nineno nobody non none noone nor normally not nothing novel now nowhere o obviously of off often oh ok okay old on once one ones only onto or other others otherwise ought our ours ourselves out outside over overall own p particular particularly per perhaps placed please plus possible presumably probably provides q que quite qv r rather rd re really reasonably regarding regardless regards relatively respectively right ssaid same saw say saying says second secondly see seeing seem seemed seeming seems seen self selves sensible sent serious seriously seven several shall she should shouldn't since six so some somebody somehow someone something sometime sometimes somewhat somewhere soon sorry specified specify specifying still sub such sup sure t t's take taken tell tends th than thank thanks thanx that that's thats the their theirs them themselves then thence there there's thereafter thereby therefore therein theres thereupon these they they'd they'll they're they've think third this thorough thoroughly those though three through throughout thru thus to together too took toward towards tried tries truly try trying twice two u un under unfortunately unless unlikely until unto up upon us use used useful uses using usually uucp v value various very via viz vs w want wants was wasn't way we we'd we'll we're we've welcome well went were weren't what what's whatever when whence whenever where where's whereafter whereas whereby wherein whereupon wherever whether which while whither who who's whoever whole whom whose why will willing wish with within without won't wonder would would wouldn't x y yes yet you you'd you'll you're you've your yours yourself yourselves z zero

Anexo 2: Lista de palabras vacías de la lengua española

él ésta éstas éste éstos última últimas último últimos a añadió aún actualmente adelante además afirmó agregó ahí ahora al algún algo alguna algunas alguno algunos alrededor ambos ante anterior antes apenas aproximadamente aquí así aseguró aunque ayer bajo bien buen buena buenas bueno buenos cómo cada casi cerca cierto cinco comentó como con conocer consideró considera contra cosas creo cual cuales cualquier cuando cuanto cuatro cuenta da dado dan dar de debe deben debido decir dejó del demás dentro desde después dice dicen dicho dieron diferente diferentes dijeron dijo dio donde dos durante e ejemplo el ella ellas ello ellos embargo en encuentra entonces entre era eran es esa esas ese eso esos está están esta estaba estaban estamos estar estará estas este esto estos estoy estuvo ex existe existen explicó expresó fin fue fuera fueron gran grandes ha había habían haber habrá hace hacen hacer hacerlo hacia haciendo han hasta hay haya he hecho hemos hicieron hizo hoy hubo igual incluso indicó informó junto la lado las le les llegó lleva llevar lo los luego lugar más manera manifestó mayor me mediante mejor mencionó menos mi mientras misma mismas mismo mismos momento mucha muchas mucho muchos muy nada nadie ni ningún ninguna ningunas ninguno ningunos no nos nosotras nosotros nuestra nuestras nuestro nuestros nueva nuevas nuevo nuevos nunca o ocho otra otras otro otros para parece parte partir pasada pasado pero pesar poca pocas poco pocos podemos podrá podrán podría podrían poner por porque posible próximo próximos primer primera primero primeros principalmente propia propias propio propios pudo pueda puede pueden pues qué que quedó queremos quién quien quienes quiere realizó realizado realizar respecto sí sólo se señaló sea sean según segunda segundo seis ser será serán sería si sido siempre siendo siete sigue siguiente sin sino sobre sola solamente solas solo solos son su sus tal también tampoco tan tanto tenía tendrá tendrán tenemos tener tenga tengo tenido tercera tiene tienen toda todas todavía todo todos total tras trata través tres tuvo un una unas uno unos usted va vamos van varias varios veces ver vez y ya yo

Anexo 3 Colocaciones más frecuentes del idioma inglés, extraidas de un Corpus basado en el diario New York Times.

Extraido del libro Foundations of Natural Language Processing, cap. 5. C. Manning y H. Schütze (1999)

Tabla original, sin filtrado alguno en base a función lingüística de los términos:

Frequencia	Palabra 1	Palabra 2
80871	of	the
58841	in	the
26430	to	the
21842	on	the
21839	for	the
18568	and	the
16121	that	the
15630	at	the
15494	to	be
13899	in	a
13689	of	a
13361	by	the
13183	with	the
12622	from	the
11428	New	York
10007	he	said
9775	as	a
9231	is	a
8753	has	been
8573	for	a

En la siguiente tabla pueden verse las colocaciones más populares del mismo corpus, luego de haber aplicado un proceso de filtrado de palabras vacias.

Frequencia	Palabra 1	Palabra 2	Función linguística
11487	New	York	A N
7261	United	States	A N
5412	Los	Angeles	N N
3301	last	year	A N
3191	Saudi	Arabia	N N
2699	last	week	A N
2514	vice	president	A N
2378	Persian	Gulf	A N
2161	San	Francisco	N N
2106	President	Bush	N N
2001	Middle	East	A N
1942	Saddam	Hussein	N N
1867	Soviet	Union	A N
1850	White	House	A N

1633	United	Nations	A N
1337	York	City	N N
1328	oil	prices	N N
1210	next	year	A N
1074	chief	executive	A N
1073	real	estate	A N

Nota: función linguística A adjetivo y N sustantivo

Anexo 4 Ejemplo de la ley de Zipf sobre un texto en español

Procesamiento de términos simples del libro "*La triste Historia de la Cándida Eréndira y su Abuela Desalmada*" de Gabriel García Márquez.

Extracto de términos ordenado por frecuencia

Término	F	Término	F	Término	F
La	590	cuando	46	grito	15
De	534	como	44	pregunto	15
El	361	hasta	36	tan	15
Y	264	estaba	35	voz	15
Que	252	si	32	casa	14
En	226	sin	31	esta	14
A	209	me	30	mientras	14
Se	170	sus	30	nadie	14
Con	166	entonces	29	tu	14
No	158	era	28	desierto	13
Los	149	ella	22	holandes	13
Dijo	147	fotografo	21	manana	13
Abuela	138	ojos	21	mundo	13
Erendira	127	hizo	19	muy	13
Un	123	mano	18	otra	13
Le	122	solo	18	vez	13
Una	114	vio	18	contesto	12
Del	97	yo	18	ni	12
Su	91	cama	17	pesos	12
Ulises	87	carpa	17	son	12
Por	86	dos	17	todo	12
Las	76	fue	17	antes	11
Lo	75	mi	17	asi	11
Para	71	noche	17	conductor	11
Al	70	porque	17	contra	11
Pero	61	esa	16	desde	11
Es	58	hombre	16	dio	11
Habia	47	viento	16	donde	11
Mas	47	volvio	16	dormida	11
Te	47	ya	16	grande	11

Vocabulario: 2548 términos
Hapax legonemas: 1677 (65.7 %) (términos con 1 sola ocurrencia)

Cálculo de la constante C para cada término

Orden	Término	Ocurrencias	Frecuencia Rel (Ocurrencia/N)	C=orden*frecuencia
1	la	590	0,057521693	0,057521693
2	de	534	0,052062006	0,104124013
3	el	361	0,035195476	0,105586429
4	y	264	0,02573852	0,10295408
5	que	252	0,024568587	0,122842937
6	en	226	0,022033733	0,132202398

7	a	209	0,020376328	0,142634299
8	se	170	0,016574047	0,132592376
9	con	166	0,016184069	0,145656625
10	no	158	0,015404114	0,154041143
11	los	149	0,014526665	0,159793312
12	dijo	147	0,014331676	0,171980111
13	abuela	138	0,013454226	0,174904943
14	erendira	127	0,012381788	0,173345033
15	un	123	0,01199181	0,179877157
16	le	122	0,011894316	0,190309057
17	una	114	0,011114361	0,188944136
18	del	97	0,009456956	0,170225212
19	su	91	0,00887199	0,168567807
20	ulises	87	0,008482012	0,169640246
21	por	86	0,008384518	0,176074876
22	las	76	0,007409574	0,163010627
23	lo	75	0,00731208	0,16817783
24	para	71	0,006922102	0,166130447
25	al	70	0,006824608	0,17061519
26	pero	61	0,005947158	0,154626109
27	es	58	0,005654675	0,152676221
28	habia, mas, te	47	0,004582237	0,128302623
29	cuando	46	0,004484742	0,130057522
30	como	44	0,004289753	0,1286926

N = Numero total de términos (se incluyen repeticiones), en el ejemplo N = 10257

Ahora es posible calcular el orden de una palabra conocida su cantidad de ocurrencias en un corpus

R n = C * (N / n) Generalmente se asume el valor de la constante en 0.1 para el inglés

Por ejemplo: ¿Cuál es el número de orden que le corresponde a una palabra que posee 70 ocurrencias para un corpus con 10000 términos?

R (70) = 0.14 * (10000 / 70)

R (70) = 20

Se concluye que una palabra con 70 ocurrencias sobre un corpus de 10000 términos estaría en el orden 20

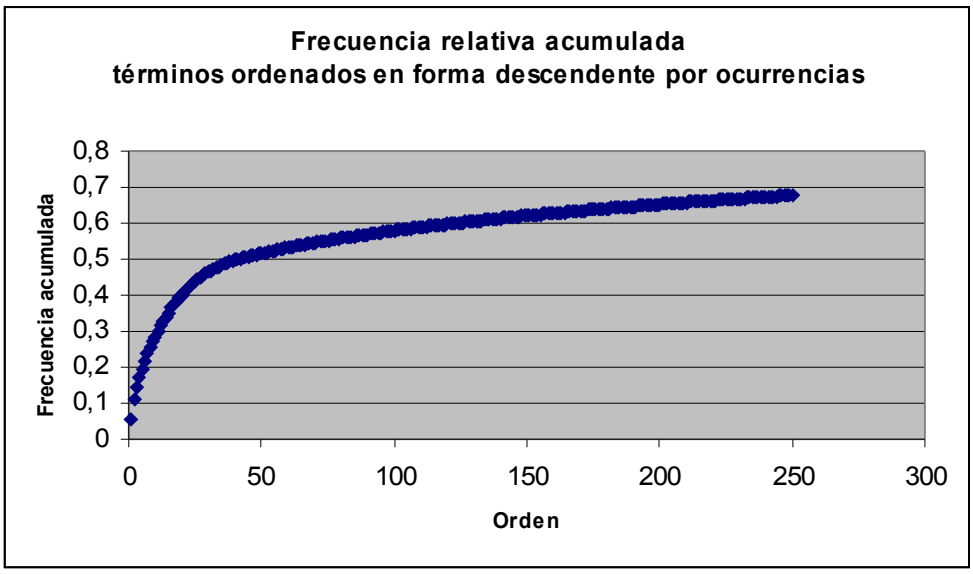
La siguiente tabla muestra la frecuencia acumulada de los términos del texto en análisis.

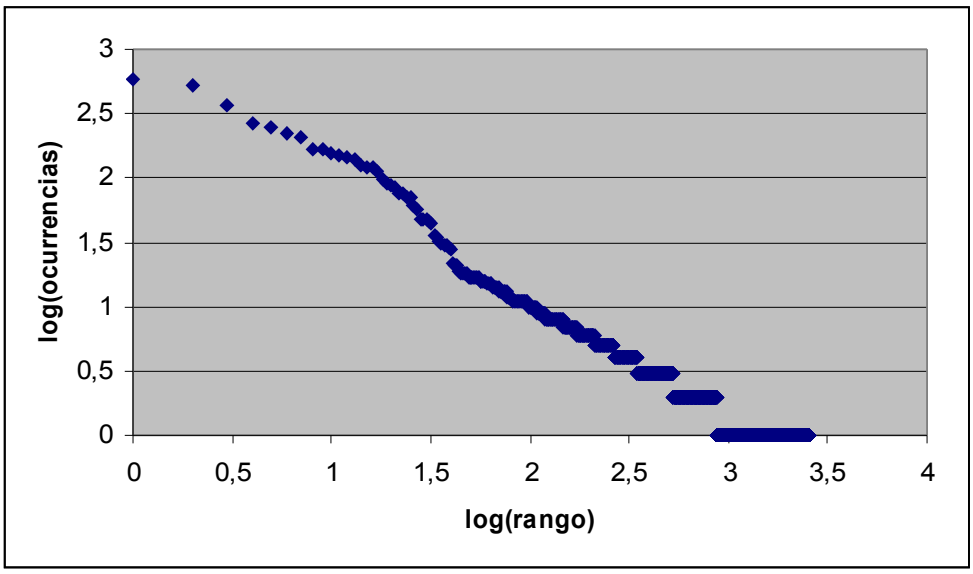
Orden	Término	Ocurrencias	Frecuencia relativa	Frecuencia acumulada
1	**la**	590	0,057522	0,057522
2	**de**	534	0,052062	0,109584
3	**el**	361	0,035195	0,144779
4	**y**	264	0,025739	0,170518

5	**que**	252	0,024569	0,195086
6	**en**	226	0,022034	0,217120
7	**a**	209	0,020376	0,237496
8	**se**	170	0,016574	0,254070
9	**con**	166	0,016184	0,270254
10	**no**	158	0,015404	0,285659
11	**los**	149	0,014527	0,300185
12	dijo	147	0,014332	0,314517
13	abuela	138	0,013454	0,327971
14	erendira	127	0,012382	0,340353
15	**un**	123	0,011992	0,352345
16	**le**	122	0,011894	0,364239
17	**una**	114	0,011114	0,375353
18	**del**	97	0,009457	0,384810
19	**su**	91	0,008872	0,393682
20	ulises	87	0,008482	0,402164
21	**por**	86	0,008385	0,410549
22	**las**	76	0,007410	0,417958
23	**lo**	75	0,007312	0,425271
24	**para**	71	0,006922	0,432193
25	**al**	70	0,006825	0,439017
26	**pero**	61	0,005947	0,444964
27	**es**	58	0,005655	0,450619
28	habia	47	0,004582	0,455201
29	**mas**	47	0,004582	0,459784
30	**te**	47	0,004582	0,464366
31	**cuando**	46	0,004485	0,468851
32	**como**	44	0,004290	0,473140
33	**hasta**	36	0,003510	0,476650
34	estaba	35	0,003412	0,480062
35	**si**	32	0,003120	0,483182
36	**sin**	31	0,003022	0,486205
37	**me**	30	0,002925	0,489129
38	**sus**	30	0,002925	0,492054
39	**entonces**	29	0,002827	0,494882
40	era	28	0,002730	0,497611
41	**ella**	22	0,002145	0,499756
42	fotografo	21	0,002047	0,501804

En negrita se indicaron las denominadas palabras comunes o vacías, que son mayoría en el conjunto presentado.

Nótese que la suma de las ocurrencias 42 términos acumulan el 50% de la cantidad total de ocurrencias de los 2506 términos restantes. Tal relación se observa en el siguiente gráfico:





Representación del logaritmo del rango versus el logaritmo de las ocurrencias. Por resultado se observa siempre una curva con pendiente –1, en las puntas la ley generalmente no se cumple.

Orden	Término	Ocurren-cias	Frecuencia relativa	Frecuencia acumulada	bytes	bytes totales	bytes acumulados	Frecuencia relativa bytes	Frecuencia acumulada bytes
1	la	590	0,057522	0,057522	2	1180	1180	0,025878	0,025878
2	de	534	0,052062	0,109584	2	1068	2248	0,023422	0,049299
3	el	361	0,035195	0,144779	2	722	2970	0,015834	0,065133
4	y	264	0,025739	0,170518	1	264	3234	0,005790	0,070923
5	que	252	0,024569	0,195086	3	756	3990	0,016579	0,087502
6	en	226	0,022034	0,217120	2	452	4442	0,009912	0,097414
7	a	209	0,020376	0,237496	1	209	4651	0,004583	0,101998
8	se	170	0,016574	0,254070	2	340	4991	0,007456	0,109454
9	con	166	0,016184	0,270254	3	498	5489	0,010921	0,120375
10	no	158	0,015404	0,285659	2	316	5805	0,006930	0,127305

11	los	149	0,014527	0,300185	3	447	6252	0,009803	0,137108
12	dijo	147	0,014332	0,314517	4	588	6840	0,012895	0,150003
13	abuela	138	0,013454	0,327971	6	828	7668	0,018158	0,168162
14	erendira	127	0,012382	0,340353	8	1016	8684	0,022281	0,190443
15	un	123	0,011992	0,352345	2	246	8930	0,005395	0,195838
16	le	122	0,011894	0,364239	2	244	9174	0,005351	0,201189
17	una	114	0,011114	0,375353	3	342	9516	0,007500	0,208689
18	del	97	0,009457	0,384810	3	291	9807	0,006382	0,215071
19	su	91	0,008872	0,393682	2	182	9989	0,003991	0,219062
20	ulises	87	0,008482	0,402164	6	522	10511	0,011448	0,230509
21	por	86	0,008385	0,410549	3	258	10769	0,005658	0,236167
22	las	76	0,007410	0,417958	3	228	10997	0,005000	0,241168
23	lo	75	0,007312	0,425271	2	150	11147	0,003290	0,244457
24	para	71	0,006922	0,432193	4	284	11431	0,006228	0,250685
25	al	70	0,006825	0,439017	2	140	11571	0,003070	0,253756
26	pero	61	0,005947	0,444964	4	244	11815	0,005351	0,259107
27	es	58	0,005655	0,450619	2	116	11931	0,002544	0,261650
28	habia	47	0,004582	0,455201	5	235	12166	0,005154	0,266804
29	mas	47	0,004582	0,459784	3	141	12307	0,003092	0,269896
30	te	47	0,004582	0,464366	2	94	12401	0,002061	0,271958
31	cuando	46	0,004485	0,468851	6	276	12677	0,006053	0,278010
32	como	44	0,004290	0,473140	4	176	12853	0,003860	0,281870
33	hasta	36	0,003510	0,476650	5	180	13033	0,003947	0,285818
34	estaba	35	0,003412	0,480062	6	210	13243	0,004605	0,290423
35	si	32	0,003120	0,483182	2	64	13307	0,001404	0,291827
36	sin	31	0,003022	0,486205	3	93	13400	0,002040	0,293866
37	me	30	0,002925	0,489129	2	60	13460	0,001316	0,295182
38	sus	30	0,002925	0,492054	3	90	13550	0,001974	0,297156
39	entonces	29	0,002827	0,494882	8	232	13782	0,005088	0,302243
40	era	28	0,002730	0,497611	3	84	13866	0,001842	0,304086
41	ella	22	0,002145	0,499756	4	88	13954	0,001930	0,306015
42	fotografo	21	0,002047	0,501804	9	189	14143	0,004145	0,310160

Cantidad total de términos únicos = 2547
Cantidad total de términos con repeticiones = 10257
Cantidad total de bytes que suman todos los términos = 45599

Nótese en la tabla anterior que la frecuencia acumulada de las ocurrencias no es la misma que la frecuencia acumulada de los bytes, eso se da por que en general las palabras vacías son más cortas que el resto de las palabras.

Anexo 5: Búsquedas en Texto Completo con el motor de base de datos MySQL

La herramienta de búsqueda en texto completo del motor de base de datos MySQL brinda la posibilidad de incorporar objetos de texto (documentos) dentro de un campo de una tabla de la base de datos. De esta manera, se soportan funciones de búsqueda equivalentes a las de un sistema de recuperación de información.

Este motor de base de datos permite búsquedas booleanas y – además – búsquedas por similitud entre la consulta y los documentos, lo que posibilita realizar un ranking de la respuesta. Estas capacidades de búsqueda en texto completo son una alternativa que extienden la funcionalidad del motor de base de datos y permiten la búsqueda sobre campos cuyo contenido no tiene una estructura determinada.

Para implementar esta funcionalidad se utilizan índices especiales. Los índices para una búsqueda Full-Text son inicializados sobre campos específicos de las tablas y solamente pueden ser implementados sobre tablas tipo MyISAM (que son las tablas por defecto del SGBD). Los tipos de datos de los campos que se pueden utilizar para definir índices full-text son char, varchar y text. Dichos índices se definen al momento de crear una tabla aunque también se pueden agregar luego. Para mayor información puede consultar a Zawodny[19], Harper[20] y la la documentación oficial del desarrollador[21].

A los efectos de ilustrar el funcionamiento de esta capacidad se presenta un ejemplo completo de su implementación. Suponga que se tiene una tabla para almacenar productos, sobre la cual se definen dos campos full-text para luego poder realizar búsquedas sobre éstos. La creación de la tabla se realiza de la siguiente manera:

```
CREATE TABLE PRODUCTOS
(
     id             int unsigned not null primary key,
     título         varchar(30),
     descripción    text,
     url            varchar (100),
     precio         real,
     fulltext (título, descripción)
);
```

Otra posibilidad es definir índices por separado para cada campo por el cual se quiere indexar por texto completo. La diferencia radica en que las consultas luego se realizan sobre un campo en particular (y no ambos). Por ejemplo, en vez de especificar:

```
fulltext (título, descripción)
```

se puede especificar:

[19] Zawodny, J. "*MySQL Full-Text Search Rocks My World*". Jeremy Zawodny's blog. Marzo, 2003. http://jeremy.zawodny.com/blog/archives/000576.html

[20] Harper, M. "*Getting Started With MySQL's Full-Text Search Capabilities*". Agosto, 2002. http://www.devarticles.com/art/1/195

[21] *MySQL Full-text Search*. MySQL Reference Manual. 2003. http://www.mysql.com/doc/en/Fulltext_Search.html

```
fulltext (título),
fulltext (descripción)
```

lo que genera que el motor de base de datos cree dos índices diferentes por cada campo. La decisión de indexar por uno ó varios campos juntos ó bien por separado tiene que ver con el objetivo final que se persigue en la implementación y el tipo de consultas que se van a realizar.

El uso de la cláusula **fulltext** dentro de la declaración de la tabla indica que se realice la indexación de los campos pasados como argumento (en este caso ejemplo, corresponde a los campos título y descripción).

En muchos casos se tienen bases de datos con tablas existentes (de muchos registros), quizás creadas con versiones anteriores de MySQL. Si se requiere utilizar esta nueva la capacidad se pueden modificar las tablas utilizado el comando SQL ALTER TABLE, por ejemplo:

```
ALTER TABLE productos ADD fulltext (título, descripción);
```

Búsquedas

Las búsquedas (consultas) se realizan a partir de dos nuevas funciones agregadas a la sentencia SQL SELECT denominadas MATCH() y AGAINST(). La función MATCH() permite realizar una búsqueda sobre una o más columnas incluidas en el índice fulltext, mientras que la condición consiste en una cadena de una ó más palabras ingresada como argumento de la función AGAINST().

Siguiendo con el ejemplo anterior, se solicita una consulta por “cloruro proteínas fibras” sobre el índice de texto:

```
SELECT título FROM productos WHERE MATCH(descripción) AGAINST ("cloruro fibra proteína")
```

Respuesta:

```
+-------------+
| título      |
+-------------+
| arroz       |
| gelatina    |
| mermelada   |
+-------------+
```

Las búsquedas no son sensitivas a mayúsculas, por lo que una consulta con la condición **AGAINST** (“CLORURO Fibra proteína”) entrega el mismo resultado.

Como se puede observar, MySQL devuelve documentos que se “aproximan” a la consulta realizada. Igual resultado se hubiera obtenido si la consulta hubiera sido “el cloruro, la proteína y la fibra” ya el motor de base de datos elimina palabras vacías.

La intención al proveer índices de texto completo es que se puedan realizar

búsquedas sobre el contenido de documentos escritos en lenguaje natural, que son incorporados a alguna tabla de una base de datos como campos tipo text, char ó varchar (Téngase en cuenta que cuando se mencione documento, se está haciendo referencia a un texto en un campo textual de una tabla). Las características principales de la construcción de los índices y de las búsquedas en texto completo son:

- Se excluyen las palabras de menos de 4 caracteres
- Se excluyen las palabras vacías
- Las palabras separadas por guiones son tratadas como dos términos diferentes
- Se excluyen palabras parciales, es decir las búsquedas se realizan por las palabras completas
- Si una palabra se encuentra en más del 50% de los registros buscados, entonces esos registros no son devueltos. Este es un umbral a partir del cual se consideran registros relevantes
- Los resultados se ordenan por relevancia

Orden por Relevancia

Respecto del orden en que son devueltos los resultados, para cada fila en la tabla la función MATCH retorna un valor de relevancia. Dicho valor es una medida de similitud entre la cadena de búsqueda (argumento de la función AGAINST()) y el texto en la columna que se le dio como parámetro a la función MATCH.

Cuando se utiliza MATCH en la cláusula WHERE las filas retornadas son ordenadas automáticamente de manera descendente, debido a que el primer resultado es el más “aproximado” ó más “cercano” a la consulta. Los valores de relevancia devueltos son números de punto flotante no negativos. Por ejemplo, la consulta:

```
SELECT título, MATCH(título, descripción) relevancia AGAINST ("cloruro fibra proteína") FROM productos WHERE MATCH (título, descripción) AGAINST ("cloruro fibra proteína")
```

retorna:

```
_____________________________________
 título         relevancia
_____________________________________
arroz           2.8396340442348
gelatina        1.4287065109334
mermelada       0.60909244387936
_____________________________________
```

donde se observa el grado de similitud o relevancia calculado por MySQL entre el título y descripción del producto y la consulta.

La relevancia se computa de acuerdo al numero de palabras en la fila, el número de palabras únicas en la fila, el numero total de palabras en la colección de documentos (filas) y el numero de documentos (filas) que contienen una palabra en particular.

El modo de asignación de pesos a los términos se realiza teniendo en cuenta los

siguientes parámetros: a cada palabra correcta (que cumpla con las condiciones antes mencionadas) tanto en la consulta como en la colección de documentos se le asigna un peso según su significado dentro de los mismos. El peso asignado varía según algunas consideraciones, a saber: una palabra que se encuentre en muchos documentos tendrá un valor bajo como peso (inclusive 0) porque se supone que es poco su valor semántico en dicha colección. De esta manera, si la palabra aparece pocas veces en un determinado documento tendrá un alto peso ya que se supone que esa palabra es representativa del mismo. Los pesos de las palabras son combinados para calcular la relevancia.

Cabe destacar que esta técnica trabaja mejor con colección con gran cantidad de documentos. Una búsqueda por solo una palabra puede no producir resultados, esto es si esta se encuentra presente en la mitad de las filas de la tabla (ó en la mitad de los documentos), es tratada como una palabra vacía. Una palabra que se encuentre en la mitad de las filas es probablemente menos relevante para los documentos almacenados. Supóngase el siguiente ejemplo:

```
SELECT título, MATCH(título, descripción) relevancia AGAINST ("ácido") FROM productos WHERE MATCH (título, descripción) AGAINST ("ácido")
```

La consulta no retorna resultados ya que – como se mencionó anteriormente – la palabra se encuentra en la mitad de los documentos.

Búsquedas Booleanas

Si bien las búsquedas en texto completo pueden permitir la construcción de herramientas más flexibles, MySQL incorpora – a partir de la versión 4 – la posibilidad de realizar búsquedas booleanas. Esta característica aumenta el poder de recuperación ya que permite "evitar" las restricciones impuestas por la indexación de texto completo.

Para utilizar esta característica se agrega a la función AGAINST el modificador IN BOOLEAN MODE, por ejemplo:

```
SELECT título FROM productos WHERE MATCH (título, descripción) AGAINST ("cloruro" IN BOOLEAN MODE)
```

La diferencia fundamental con la forma normal full-text está en que a las búsquedas no se les aplica el límite del 50%, por lo que las palabras pueden aparecer en menos de la mitad de filas. A continuación se muestra una tabla con alguno de los operadores booleanos disponibles (por la descripción completa consultar la guía oficial de MySQL).

+	Exige que la palabra se encuentre en el campo de búsqueda. Ejemplo: `SELECT título FROM productos WHERE MATCH (descripción) AGAINST ("+cloruro" IN BOOLEAN MODE)`
-	Exige que la palabra no se encuentre en el campo de búsqueda. Ejemplo: `SELECT título FROM productos WHERE MATCH (descripción) AGAINST ("-café" IN BOOLEAN MODE)`

*	Comodín que indica cero ó más caracteres. Solo válido al final de la palabra de la expresión de consulta. Ejemplo: **SELECT** título **FROM** productos **WHERE MATCH** (descripción) **AGAINST** ("limón*" **IN BOOLEAN MODE**)
"	El texto encerrado entre comillas se toma como frase. Ejemplo: **SELECT** título **FROM** productos **WHERE MATCH** (descripción) **AGAINST** ("ají morrón" **IN BOOLEAN MODE**)

Ejemplo de un motor de indexación y consulta simple

Con el fin de mostrar una posible aplicación de las características de búsqueda en texto completo de MySQL se escribió un motor de indexación y consulta muy simple, utilizando el lenguaje Perl (denominado sse.pl) y la interfase DBI (como API con el motor de base de datos). El motor soporta solo dos operaciones:

a) Insertar registros en una tabla de una base de datos, la cual se encuentra indexada en texto completo por dos campos.

b) Realizar consultas sobre la misma, mostrando los resultados y la relevancia de cada uno de los mismos respecto de la expresión de consulta.

La tabla dentro de la base de datos recibe el nombre `docs` y su estructura se detalla a continuación.

Tabla `docs`

```
id              smallint
título          varchar(200)
autor           varchar(50)
texto           text
```

Los campos sobre los que se especificaron índices son título y texto. Dichos índices se especificaron por separado, por lo que las consultas se realizan sobre un campo u otro (no ambos simultáneamente). Entonces, la sintaxis completa de creación de la tabla resulta:

```
CREATE TABLE docs
(
     id             smallint unsigned not null primary key,
     título         varchar(200),
     autor          varchar(50),
     texto          text,
     fulltext (título),
     fulltext (texto)
);
```

Luego, las operaciones soportadas por el motor de indexación y búsqueda

requieren de los siguientes parámetros:

Para insertar un registro nuevo:

```
perl sse.pl –i id título autor nombre_archivo_texto
```

Donde

`id`	Es un identificador de registro
`título`	Es el título de la nota ó noticia
`autor`	Corresponde al autor ó creador de la misma
`nombre_archivo_texto`	Es el nombre del archivo de texto que el motor va a leer para insertar en el campo texto de la tabla (puede incluir el path)

Por ejemplo:

```
perl sse.pl –i 124 "Búsqueda en MySQL" "gabriel" "./notas/nota124.txt"
```

Para realizar una consulta:

```
perl sse.pl –q indice termino_1 termino_2 … termino_n
```

Donde

`indice`	Corresponde a uno de los dos índices posible, es decir, título ó texto
`término_1… término_n`	Son las palabras utilizadas para la búsqueda

Por ejemplo:

```
perl sse.pl –q título auto carrera
```

Mediante esta sintaxis se buscarán los registros más relevantes para palabras `auto` y `carrera` según el campo `título`, mientras que:

```
perl sse.pl –q texto mascota alimento cachorro
```

realizará una búsqueda por los registros más relevantes de acuerdo a las palabras `mascota`, `alimento` y `cachorro` según el campo `texto`.

A continuación, se muestra el listado completo del código fuente el Perl del motor de indexación y búsqueda (sse.pl).

`See.pl`

```
#!/usr/bin/perl
use DBI;

my($i, $query);
```

```
my($drh, $dbh, $qyx, $rsp);
my($command, $idd, $tit, $aut, $txt);

$drh=DBI->install_driver("mysql");
$dbh=DBI->connect('dbi:mysql:database=test', 'root', '');

$command = $ARGV[0];

if ($command eq "-i")
{
      $idd = $ARGV[1];  # id_documento
      $tit = $ARGV[2];  # título
      $aut = $ARGV[3];  # autor
      $txt = $ARGV[4];  # nombre del archivo con el texto

      if ($idd eq "" || $tit eq "" || $aut eq "" || $txt eq "")
      {
            &print_help;
      }
      else
      {
            &insert ($idd, $tit, $aut, $txt);
      }
}
elsif ($command eq "-q")
{
      $i = 2;
      $terms = "";
      $ndx = $ARGV[1];

      while ($ARGV[$i] ne "")
      {
          $terms = $terms . "\t" . $ARGV[$i];
          $i++;
      }


      if (($ndx ne "título" && $ndx ne "texto") ||  $terms eq "")
      {
            &print_help;
      }
      else
      {
            &query ($ndx, $terms);
      }

}
else
{
      $print_help;
}

exit(0);

sub print_help
{
      print "\n";
      print "usar: perl se.pl [opcion] [argumentos] \n";
      print "\n";
      print "donde [opcion]: \n\n";
      print "  -i \t id_doc título autor nombre_archivo_texto \n";
```

```
	print "\n";
	print "  -q \t [título|texto] termino_1 termino_2 ... termino_n \n";
	print "\n";
}

sub insert
{
	my($idd, $tit, $aut, $txt) = @_;

	$texto = "";

	open (IN, "<$txt");
	while (<IN>)
	{
		$texto = $texto . $_;
	};
	close(IN);

	print	"$idd \t  $tit \t $aut \t $texto \n";

	$query = "INSERT INTO docs VALUES ($idd, '$tit', '$aut', '$texto')";

	$qyx = $dbh->prepare($query);
	$rsp = $qyx->execute();
}

sub query
{
	my($ndx, $terms) = @_;

	$cond = "match($ndx) against('". $terms . "')";
	$query = "SELECT id, título, autor, $cond  FROM docs WHERE $cond ";

	#print "\n $query \n";

	print "\n\n";
	print
"-------------------------------------------------------------\n";
	print "id \t\t título \t\t autor \t\t relevancia\n";
	print
"-------------------------------------------------------------\n";

	$qyx = $dbh->prepare($query);
	$rsp = $qyx->execute();

	while (@tupla = $qyx->fetchrow())
	{
		for ($i=0; $i < 4; $i++)
		{
			print $tupla[$i] . "\t\t";
		}
		print "\n";
	}
	print
"-------------------------------------------------------------\n";
	print "\n";
}
```