Recuperación de Pasajes Multilingüe para la Búsqueda de Respuestas

José Manuel Gómez Soriano

Director
Dr. Emilio Sanchis Arnal

Tesis Doctoral

A mi madre y amigos

Agradecimientos

Realizar una tesis es un esfuerzo considerable, no sólo para mí sino para muchas otra gente que ha colaborado directa o indirectamente en mi trabajo. Por ello quiero agradecer a Emilio Sanchis, mi director de tesis, que me ha dado todo su apoyo. Por su esfuerzo y dedicación; por sus sabios consejos que me han permitido orientarme sabiamente en el área de mi investigación y me han permitido cruzar el charco para tener la suerte de trabajar con Manuel Montes, un experto en Búsqueda de Respuestas; y por su trabajo burocrático, una labor que nunca está reconocida pero es de vital importancia para la investigación y me ha permitido tener financiación en todos mis años de doctorado.

La segunda persona a la que me gustaría dar las gracias es a Paolo Rosso que, aunque me ha hecho trabajar muy intensamente, a permitido que pueda terminar la tesis en tan poco tiempo. Muchas gracias por sus pacientes y concienzudas lecturas de mis trabajos. Por sus anotaciones que han sabido rectificar mis artículos con gran acierto e inteligencia. Y por sus esfuerzos más allá de lo que se exige a un profesor y compañero. Gracias por la oportunidad que me brindó de viajar a la India e investigar en dos prestigiosos institutos de aquel país durante más de dos meses.

A Manuel Montes. Por su atención durante mi estancia en México, por sus consejos y porque ha sido la persona que me ha inspirado en esta investigación y me ha guiado acertadamente en mi trabajo. Ha sido de gran ayuda tanto como asesor como compañero y ha realizado un esfuerzo considerable para mejorar mi estancia en Heroica Puebla de Zaragoza, en México. A Luis Villaseñor Pineda, Rafa Guzmán, Alberto Téllez y José María Cañas por su amistad en aquellas lejanas tierras.

A Davide Buscaldi, mi compañero de fatiga, y al que espero ver pronto escribiendo su tesis. Por tu tiempo y ayuda a la hora de realizar mis

experimentos y por su colaboración en tantos proyectos. Por sus esfuerzos y por su gran entrega que ha hecho posible que nuestro sistema participara en los concursos internacionales del Cross Language Evaluation Forum 2005 y 2006.

Un rinconcito de mi corazón siempre estará reservado para Atiur Rahman Khan, por desvelarse tan desinteresadamente durante mi estancia en la India y por su inestimable amistad y compañía. A Doctor Raymond, Mahesh Kulkarni, Bhanu Prasad y Vasudema Varma, por su amable recibimiento en los centros de CDAC de Pune y Hyderabad y hacer todo lo posible para que me sintiera como en casa en un país con una sociedad y una cultura tan distinta. A Kuulaa Qaqqabaa y a tantos otros amigos que conocí en la India.

Me gustaría agradecer también a todos mis compañeros de los laboratorios 3L01, 3L02 y 1L06, especialmente a Empar Bisbal y a Natasha Ponomareva. Gracias a todos por hacer amena la sufrida tarea de la investigación. A Empar por ser tan buena amiga como compañera, siempre dispuesta a echarme una mano. A Natasha por su inestimable apoyo y por su ayuda con la traducción de mis artículos al inglés.

Al personal de secretaría (Ana Gadea, Rosa Martínez, Antonio Alfani y Ricardo Sabater) por su entrega a su trabajo, por hacer nuestra labor de investigación mucho más sencilla y por su amabilidad y saber hacer.

También me gustaría agradecer a todos mis profesores de DEA por enseñarme todo lo que sé en mi área de investigación y estar siempre dispuestos a ayudarme y a aconsejarme.

Por último agradecer a todas esas personas cuyo nombres no aparecen en este agradecimiento y me han ayudado y dado su apoyo, como es el caso de mis compañeros en la Universidad Politécnica de Valencia, en el Instituto Nacional de Astrofísica de México, en el Centro para el Desarrollo de la Computación Avanzada de Pune y Hyderabad. Además, qusiera pedir disculpas a todos aquellos que por mi falta de memoria no haya mencionado en los presentes agradecimientos y que han hecho posible que yo esté aquí escribiendo estas líneas. Sinceramente, muchas gracias a todos y todas.

Índice general

1. Introducción **1**
1.1. Conceptos generales 3
1.1.1. Término 3
1.1.2. Documento 4
1.1.3. Consultas y preguntas 6
1.1.4. Indexación 7
1.1.5. Ficheros invertidos 8
1.1.6. Palabras comunes o *stopwords* 9
1.1.7. Términos relevantes o pivotes 10
1.1.8. *Stemming* y lematización 11
1.1.9. Pasaje 12
1.1.10. Expansión de la pregunta 13
1.2. Áreas de la Recuperación de Información 16
1.2.1. Recuperación de la Información *Ad hoc* 17
1.2.2. Búsqueda de Respuestas 20
1.2.3. Extracción de la Información 20
1.3. Motivación y objetivos de la tesis 21
1.4. Organización de la tesis 24

2. Estado del arte **27**
2.1. Sistemas de Recuperación de Información 27
2.1.1. Modelo booleano 27
2.1.2. Modelo Booleano Difuso 29
2.1.3. Modelo Booleano Extendido 29
2.1.4. Modelo Espacio Vectorial 31
2.2. Sistemas de Recuperación de Pasajes 33
2.3. El tamaño del pasaje 34

2.3.1. Modelos de ventanas 35
2.3.2. Modelos de discurso 36
2.3.3. Modelos semánticos 37
2.3.4. Comparativa entre modelos de división de documentos 38
2.3.5. Solapamiento 40
2.4. Sistemas de Búsqueda de Respuestas 41
2.5. Análisis de la pregunta 44
2.6. Sistemas de Recuperación de pasajes 44
2.7. Extracción de la respuesta 47
2.8. Campañas de evaluación de sistemas de Recuperación de la Información y Busqueda de Respuestas 48
2.8.1. Text REtrieval Conference 48
2.8.2. Cross-Language Evaluation Forum 52

3. Modelos de n-gramas **55**
3.1. Arquitectura del sistema 57
3.2. Modelo de N-gramas Simple 59
3.3. El modelo de n-gramas Term Weight 64
3.4. El modelo de Densidad de Distancias de N-gramas . . . 66
3.5. Reformulaciones de la pregunta 71
3.6. Filtros de pasajes 73

4. Descripción del sistema **77**
4.1. Arquitectura del sistema 77
4.2. Java Process Manager 79
4.3. Archivo de Configuración de Ejecuciones 80
4.3.1. Acciones 81
4.3.2. Ejecuciones 82
4.3.3. Procesos y subprocesos 83
4.3.4. Métodos 87
4.4. Parámetros 90
4.4.1. Parámetros estáticos 90
4.4.2. Parámetros dinámicos 92
4.4.3. Referencia entre parámetros 94
4.4.4. Array de parámetros 94
4.5. Ámbito de las acciones y parámetros 95

4.6. JAVA Database Information Retrieval 96
4.7. Acciones y parámetros condicionados 100
4.8. JAVA Information Retrieval System 103
4.8.1. Indexación de la colección de documentos 103
4.8.2. Búsqueda de pasajes local 104
4.8.3. Búsqueda de pasajes remota 105
4.8.4. Búsqueda de pasajes distribuida 107
4.8.5. Recuperación de pasajes sobre colección de preguntas 108
4.8.6. Creación de tablas de cobertura 109
4.8.7. Aplicar modelos de n-gramas a otros sistemas . . 110

5. Evaluación de JIRS **111**
5.1. Medidas de evaluación 112
5.1.1. Cobertura 112
5.1.2. Mean Reciprocal Rank 112
5.1.3. Redundancia 113
5.1.4. Precisión 114
5.2. El corpus 114
5.2.1. Clasificación de las preguntas 115
5.3. Ajustes de parámetros 119
5.3.1. Modelo Simple 120
5.3.2. Modelo Term Weight 130
5.3.3. Modelo Densidad de Distancias de N-gramas . . . 138
5.3.4. Comparación de los modelos de n-gramas 149
5.3.5. Tamaño del pasaje 157
5.4. Comparación con otros sistemas de RP 162
5.4.1. Descripción de los sistemas de RP 162
5.4.2. Evaluación de los sistemas de RP 163
5.5. Resultados en Búsqueda de Respuestas 170

6. Conclusiones **177**

Bibliografía **182**

A. Glosario de abreviaturas **201**

B. Manual de usuario **203**
B.1. Instalación 203
B.2. Modo de uso 204
B.3. Parámetros globales 205
B.4. Indexar documentos 208
B.5. Comprobar si la indexacción se ha realizado con éxito . . 213
B.6. Lanzar el motor de búsqueda de pasajes 214
B.7. Realizar una búsqueda de pasajes remota 218
B.8. Realizar una búsqueda de forma local 218
B.9. Buscar una colección de preguntas automáticamente . . . 219
B.9.1. Formato de preguntas del CLEF 219
B.9.2. Formato XML de JIRS (JIRS Questions) 220
B.9.3. Búsqueda en modo cliente/servidor 220
B.9.4. Búsqueda en modo local 222
B.10. Crear una tabla de cobertura 222
B.11. Adaptar JIRS a nuevos idiomas no soportados 225
B.12. Errores comunes 228

C. GNU General Public License **229**
C.1. Preamble 229
C.2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION 231

Índice de figuras

1.1. Los sistemas de RI permiten alcanzar la información más fácilmente 2
1.2. Ejemplo ilustrativo de la ley de Zipf 5
1.3. Cómo trata muchos sistemas de RI los documentos . . . 6
1.4. Ejemplo de un índice 7
1.5. Ejemplo de un fichero invertido 8
1.6. Pasajes 12
1.7. Ejemplo de expansión de la pregunta a partir de un thesaurus 14
1.8. Ejemplo de realimentación 15
1.9. Sistemas de Recuperación de Documentos 17
1.10. Indexación y búsqueda 19
1.11. Sistemas de Búsqueda de Respuestas 20
1.12. Extracción de la Información 21

2.1. Modelo Booleano 28
2.2. Ejemplo del modelo BE para los operadores *and* y *or* . . 30
2.3. Ejemplo del modelo EV 32
2.4. Sistemas de RP 34
2.5. Modelo de ventanas de tamaño fijo de 20 palabras 35
2.6. Modelo de discurso orientado a párrafos 36
2.7. Modelo semántico 37
2.8. Modelo de discurso orientado a frases 39
2.9. Pasajes sin solapamientos 40
2.10. Pasajes con solapamientos 41
2.11. Típica estructura de un sistema de BR con modulo de RD 42
2.12. Típica estructura de un sistema de BR con módulo de RP 43

3.1. Ejemplo de dos posibles pasajes relevantes ante una pregunta del usuario. 55
3.2. Arquitectura de JIRS para la búsqueda con modelos de n-gramas 58
3.3. Arquitectura de JIRS para los modelos de n-gramas Simple y Term Weight 60
3.4. Ejemplo de la extracción de los n-gramas de la pregunta 61
3.5. Ejemplo de extracción de los n-gramas en frases 62
3.6. Ejemplo del modelo de n-gramas Simple 64
3.7. Ejemplo del modelo de n-gramas Term Weight 65
3.8. Arquitectura del modelo de Densidad de Distancias de N-gramas 68
3.9. Ejemplo del modelo de Densidad de Distancias de N-gramas 69
3.10. Arquitectura de JIRS con reformulación de la pregunta . 71
3.11. Arquitectura de JIRS con filtros y el múdlo de reformulación opcional 73

4.1. Arquitectura general de JIRS 78
4.2. Ejemplo de una ejecución mediante el JPM 80
4.3. Ejemplo de una ejecución 82
4.4. Un proceso 83
4.5. Funciones de un proceso 84
4.6. Ejemplo de subprocesos 84
4.7. Ejemplo de procesos paralelos 87
4.8. Arquitectura de los métodos 88
4.9. Intercambio de información entre dos procesos utilizando parámetros dinámicos 93
4.10. Pasos realizados para indexar la colección de documentos 104
4.11. El sistema de búsqueda local 105
4.12. El sistema de búsqueda con arquitectura cliente/servidor 106
4.13. El sistema de búsqueda con arquitectura distribuida . . . 107
4.14. Ejemplo de una búsqueda multilingüe 108
4.15. Pasos para la recuperación de pasajes sobre colección de preguntas 109

5.1. Cobertura en el 5º, 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 121

5.2. Redundancia en el 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 124
5.3. MRR en el 5ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 125
5.4. Precisión del modelo Simple a medida que aumentamos el número de pasajes examinados 126
5.5. Precisión por pasaje 126
5.6. Cobertura en el 5º, 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 131
5.7. Redundancia en el 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 133
5.8. MRR en el 5ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 134
5.9. Precisión del modelo Term Weight a medida que aumentamos el número de pasajes examinados 134
5.10. Precisión por pasaje 135
5.11. Cobertura en el 5º, 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el SVM con factores de distancia de 0.2, 0.4, 0.6, 0.8 y 1 140
5.12. Cobertura en el 5º, 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el RW-Density con factores de distancia de 0.2, 0.4, 0.6, 0.8 y 1 141
5.13. Redundancia en el 10º, 15ºy 20ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda SVM o RW-Density y con el factor de distancia k de 0.4 144
5.14. MRR en el 5ºpasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda 145
5.15. Precisión del modelo de Densidad de Distancia de N-gramas a medida que aumentamos el número de pasajes examinados 146
5.16. Precisión por pasaje 146
5.17. Comparación de la cobertura para los modelos de n-gramas con los motores SVM y RW-Denstiy. 150

5.18. Comparación de la redundancia para los modelos de n-gramas con los motores SVM y RW-Denstiy 152
5.19. Comparación de la precisión para los modelos de n-gramas con los motores SVM y RW-Denstiy 153
5.20. Precisión por pasaje para los modelos de n-gramas con los motores SVM y RW-Denstiy 153
5.21. Comparación de la cobertura para diferentes tamaños de pasaje 158
5.22. Comparación de la redundancia para diferentes tamaños de pasaje 159
5.23. Comparación de la precisión para diferentes tamaños de pasaje 160
5.24. Precisión por pasaje para los modelos de n-gramas con los motores SVM y RW-Denstiy 161
5.25. Evaluación de la cobertura de los sistemas RW-Distance, SVM-Distance, IBM, Lucene, MITRE y MULTITEXT . 163
5.26. Comparación de la redundancia entre los distintos sistemas 164
5.27. Comparación de la precisión para los distintos sistemas . 166
5.28. Precisión por pasaje para los distintos sistemas evaluados 167

B.1. Ejemplo de la ayuda en línea de la aplicación jpm 205
B.2. Modulos que participan en la búsqueda Cliente/Servidor 215

Índice de tablas

1.1. Ejemplo de términos extraídos de noticias de la Angencia EFE de los años 1994 y 1995 4
1.2. Diferencias entre consultas y preguntas 7
1.3. Ejemplo de palabras comunes o *stopwords* para el español 10
1.4. Diferencias entre lematización y stemming 11

2.1. Peso de los términos A y B con el modelo Booleano para las consultas *and* y *or* 28
2.2. Peso de los términos A y B con el modelo BE para las consultas *and* y *or* 31

5.1. Resumen de las características de las colecciones de documentos del CLEF 115
5.2. Número de preguntas del CLEF 2005 de cada tipo para cada colección 116
5.3. Número de preguntas de cada tipo 118
5.4. Cobertura por tipo de respuesta esperado para el modelo Simple con SVM 127
5.5. Cobertura por tipo de respuesta esperado para el modelo Simple con RW-Density 128
5.6. Cobertura por tipo de respuesta esperado para el modelo Term Weight con SVM 137
5.7. Cobertura por tipo de respuesta esperado para el modelo Term Weight con RW-Density 138
5.8. Cobertura por tipo de respuesta esperado para el modelo de Densidad de Distancias de N-gramas con SVM 147

5.9. Cobertura por tipo de respuesta esperado para el modelo de Densidad de Distancias de N-gramas con RW 148
5.10. Parámetros utilizados en la comparación de los sistemas: modelo de n gramas usado, motor de búsqueda (MB) previo, pasajes devueltos por el MB y la constante k usada en los modelos de Distancia. 150
5.11. Comparativa del MRR en el 5ºpasaje para los distintos sistemas basados en n-gramas 152
5.12. Cobertura por tipo de respuesta esperado para los sistemas de n-gramas con los motores SVM y RW-Density 155
5.13. Comparativa del MRR en el 5ºpasaje para distintos tamaños de pasaje 160
5.14. Comparativa del MRR en el 5ºpasaje para cada uno de los sistemas evaluados 165
5.15. Cobertura por tipo de respuesta esperado para los sistemas comparados 168
5.16. Resultados de la tarea de BR monolingüe en español . . 172
5.17. Resultados de la tarea de BR monolingüe en francés . . . 173
5.18. Resultados de la tarea de BR monolingüe en italiano . . 174
5.19. Resultados de la tarea de BR bilingüe con preguntas en inglés y la colección de documentos en español 175
5.20. Resultados de la tarea de BR bilingüe con preguntas en español y la colección de documentos en inglés 175
5.21. Resultados en las tareas monolingües en español, francés e italiano 176

Capítulo 1
Introducción

Tradicionalmente, todo el conocimiento público era almacenado y ordenado en bibliotecas. Los bibliotecarios eran los encargados de ordenar la información de tal manera que, si uno conocía el título, autor o materia de un libro, fuera fácilmente localizable. Para ello utilizaban fichas ordenadas alfabéticamente que facilitaban notablemente la búsqueda de la información. El problema se plantea cuando un usuario intenta encontrar alguna información sin conocer exactamente el título o autor de ésta. Esto exigía un exhaustivo escrutinio entre los libros de una temática dada. Parte del problema se solventó gracias a la introducción de la informática en las bibliotecas. Usando esta tecnología, el usuario es capaz de encontrar la información que desea conociendo unos pocos términos relevantes. El problema persiste si las palabras relevantes que utiliza el usuario no se encuentran en el título del libro o en la información que los bibliotecarios han almacenado en la ficha relacionada.

En nuestros días las cosas han cambiado. Las nuevas tecnologías de la información, especialmente las relacionadas con Internet, hacen que los usuarios tengan acceso a volúmenes de información inmensos, difícilmente catalogables y sin ningún orden ni estructura preestablecidos. Ya no existen títulos o pequeños resúmenes de cada documento sino un mar de éstos en diferentes formatos y lenguas. Pero la forma de llegar a dicha información es, incluso, más importante. Ya no es factible realizar una búsqueda basada en ningún orden puesto que en Internet no existe ningún bibliotecario que clasifique y ordene los contenidos. Ni siquiera se puede, muchas veces, conocer el idioma de un documento mirando en su metainformación, pues esta metainformación, que vendría a ser información extra que describe características del propio documento, no existe

o está mal introducida, ya sea por desidia del autor o por motivos más malintencionados. Además, y dada la estructura de Internet, es imposible realizar una búsqueda secuencial con un navegador. Incluso los modernos buscadores tienen numerosos problemas para moverse entre los miles de millones de documentos que se pueden acceder desde un ordenador personal. Se plantea, en consecuencia, la necesidad de utilizar sofisticados sistemas de *Recuperación de la Información* (RI) ([BYRN99, WBM94]). Los usuarios deben buscar, de forma más o menos interactiva, entre todos los documentos para encontrar aquella información más relevante a sus necesidades. Esto plantea graves problemas a los profesionales que, día tras día, se abastecen de los conocimientos de otros para obtener los datos necesarios para su trabajo. Para facilitar esta tarea nacieron los sistemas de RI que utilizando diversas técnicas consiguen, con más o menos acierto, el poner a disposición del usuario la información que necesita.

La búsqueda de información en grandes volúmenes de datos es un problema abierto y complejo que abarca diversas tecnologías de tratamiento del lenguaje: desde simples técnicas probabilísticas hasta complejos análisis semánticos. Esto ha hecho que en recientes años la tecnología de la RI se haya dividido en diferentes ramas de investigación: desde sistemas de *Extracción de la Información* (EI) de forma automática, hasta sistemas que devuelven una respuesta concreta a una pregunta en lenguaje natural, los llamados sistemas de *Búsqueda de Respuestas* (BR).

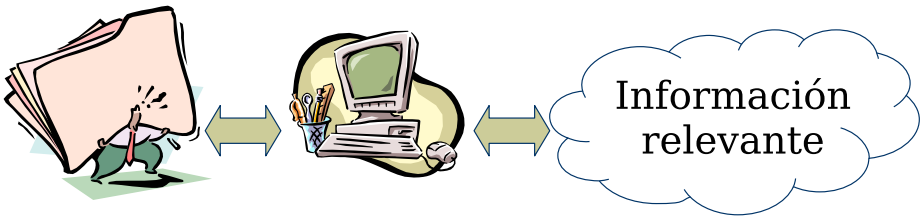


Figura 1.1: Los sistemas de RI permiten alcanzar la información más fácilmente

La mayoría de los sistemas de RI tienen dos objetivos principales: (i) obtener toda la información que pueda ser relevante para el usuario y (ii) obtener únicamente la relevante, es decir, obtener toda la información que necesita el usuario sin incluir nada más. El problema de esta tarea es que no siempre está claro qué es lo que pide el usuario. Además,

para una misma consulta varios usuarios pueden desear diferentes tipos de información. Por lo tanto, los modernos sistemas de RI no sólo se contentan en buscar las palabras claves de la consulta, sino que van más allá e intentan obtener más información por parte del usuario, incluso sin que éste se percate.

En este trabajo nos vamos a centrar en los sistemas de RI que son utilizados en BR y veremos cómo la mayoría de éstos no son tan eficaces como cabría esperar puesto que no están especialmente adaptados para dicha tarea. Veremos cuales son sus deficiencias y explicaremos los nuevos modelos que hemos desarrollado para mejorar tanto la cobertura como la precisión de estos sistemas.

Pero antes de continuar veremos unos conceptos básicos necesarios para comprender los sistemas utilizados en el área de RI.

1.1. Conceptos generales

1.1.1. Término

Un término es una secuencia de caracteres y/o números que están separados de otros términos por caracteres en blanco, signos de puntuación o caracteres especiales, aunque en idiomas aglutinativos los límites de los términos se convierte en una tarea mucho más compleja de detectar. Estos idiomas requieren de analizadores que detecten estos límites. El diseño de estos analizadores depende, en gran medida, del idioma en cuestión.

El término es la unidad elemental utilizada en RI. En muchos sistemas, cada término, ya sea de la consulta o de la colección de documentos, se relaciona con un peso que intenta medir el grado de discriminación de dicho término en la colección de documentos. Es decir, usualmente los términos con mayor peso serán aquellos que limiten más el número de documentos y, por lo tanto, serán los más importantes para realizar la búsqueda. Aunque esto depende de la función de pesado de términos y los modelos de búsqueda utilizados por el sistema de RI.

Los términos que aparecen en una colección de documentos pueden ser palabras de uno o varios idiomas, números, combinación de letras y números e, incluso, errores tipográficos y caracteres especiales. En la

Palabras	trasversal	trasladaron	peregrino	congraciar
Números	3610	350377	1994	2424229
Combinaciones	hu3003	1tambien	gl229b	16h13
Errores	viernres	españao	representaciom	informacioon

Tabla 1.1: Ejemplo de términos extraídos de noticias de la Angencia EFE de los años 1994 y 1995

tabla 1.1 se puede ver unos ejemplos reales extraídos de la *Agencia EFE* de los años 1994 y 1995.

Los sistemas de RI deben ser capaces de diferenciar los términos relevantes (véase sección 1.1.7) de aquellos que no lo son dada una consulta. Los términos que aparecen demasiadas veces pueden ser palabras comunes (ver sección 1.1.6) que no ayuden a la discriminación correcta de los documentos; y las palabras que aparecen con muy poca frecuencia son, usualmente, errores tipográficos o escritos con un particular estilo poco común. Pero existe un conjunto de palabras que no se encuentra en estos dos grupos que serán las palabras que de verdad nos interesen, es decir, los términos relevantes.

La ley de Zipf [Zip49], llamada así por el profesor de lingüística de la Universidad de Harvard, George Kingsley Zipf (1902-1950), establece una relación entre los términos que aparecen en la colección de documentos y su frecuencia de tal forma que, al ordenar dichos documentos por su frecuencia, podemos construir una gráfica como la de la figura 1.2. Esto nos permite obtener rangos que nos ayuda a eliminar las palabras no relevantes de las que sí lo son.

1.1.2. Documento

En RI, un documento es una secuencia de términos que expresan uno o más conceptos. La información que se puede obtener de los documentos no sólo proviene de las palabras que contiene sino que, además, del propio orden de estas palabras. Es muy común, entre los sistemas de RI, obviar este orden en aras de la simplificación y tratar cada documento como un conjunto de palabras inconexas. Esto produce una pérdida muy significativa de información sintáctica, semántica y contextual.

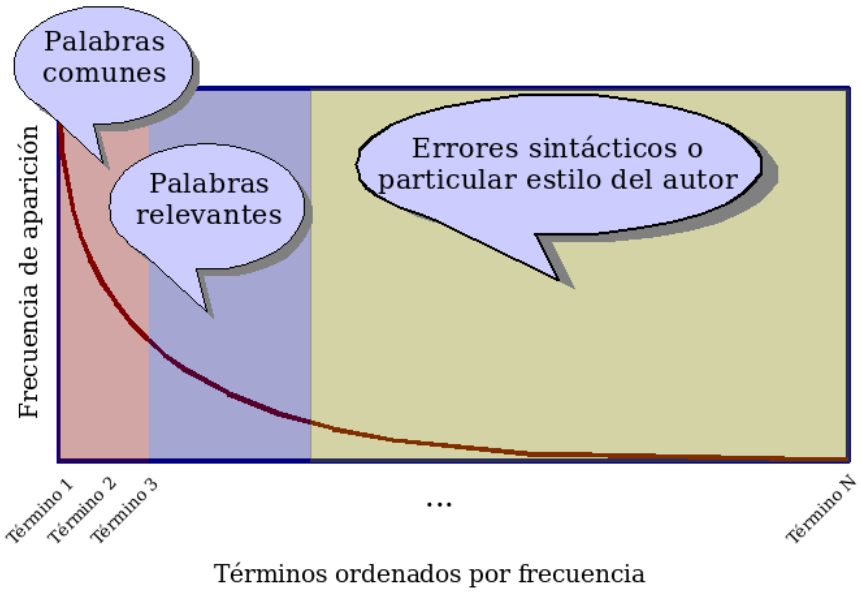


Figura 1.2: Ejemplo ilustrativo de la ley de Zipf

En el ejemplo de la figura 1.3 se puede observar la diferencia entre un texto original y un texto procesado por un sistema de RI clásico. Si intentamos entender ambos documentos, vemos que el de la derecha es incomprensible, hemos perdido, por tanto, el significado por no mantener el orden de las palabras. Además, muchos sistemas, como en el ejemplo, eliminan las palabras comunes haciendo todavía más incompresible el resultado. Si intentáramos recomponer el documento original a partir de esta bolsa de palabras podríamos obtener documentos con significados completamente distintos al original. Como vemos, los sistemas clásicos de RI trabajan con mucha menos información que la que ofrece el texto. Estos sistemas han dado muy buenos resultados en encontrar documentos relevantes a una consulta puesto que, y simplificando bastante, se puede entender que un documento relevante es aquel que habla sobre temas relacionados con los términos de la consulta. Por tanto, la información que ofrece los términos de forma inconexa es suficiente para la tarea. El problema reside al aplicar estos mismos sistemas a tareas de BR puesto que no deben encontrar documentos relevantes sino fragmentos de texto

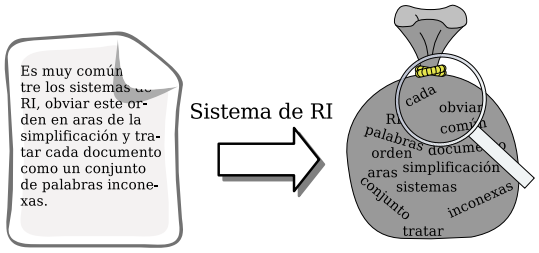


Figura 1.3: Cómo trata muchos sistemas de RI los documentos

que respondan a una pregunta. Por ello, al no tener en cuenta más que los términos relevantes, se pierde mucha información que podría ser útil para indicar si un documento o fragmento de texto responde o no a la pregunta.

1.1.3. Consultas y preguntas

Una consulta es la formulación de una necesidad de información por parte de un usuario [BYRN99]. En su forma más simple, una consulta estaría compuesta sólo por palabras claves, pero una consulta puede ser perfectamente una frase en lenguaje natural e, incluso, un documento entero. Las preguntas serían un subconjunto de las posibles consultas que se pueden realizar. Comprenden aquellas que son sintáctica y semánticamente correctas y cuyo significado se entiende como la petición de una necesidad de información.

Las consultas no tienen por qué tener una estructura sintáctica ni semántica. Sin embargo, una *pregunta* se realiza siempre en lenguaje natural y conservando un orden establecido (excepto por errores tipográficos o gramaticales) que le otorga un significado propio y muchas veces único en un contexto determinado. Las preguntas contienen mucha más información que las consultas basadas en palabras claves, puesto que éstas no pierden el significado sintáctico ni semántico de lo que realmente se pregunta y, por lo tanto, tienen menor ambigüedad.

Normalmente las consultas o preguntas son tratadas por los sistemas de RI de forma muy parecida a los documentos. Aunque el número de términos de una consulta o pregunta suelen ser mucho menor. Esto puede

Consulta	Pregunta
presidente españa	¿Quién es el Presidente de España?
capital croacia	¿Cuál es la capital de Croacia?
Robbie Williams grupo	¿Con qué grupo ha cantado Robbie Williams?

Tabla 1.2: Diferencias entre consultas y preguntas

generar problemas a algunos modelos de RI a la hora de compararlas con los documentos y es necesario, por consiguiente, normalizar o reducir los documentos para equipararlos a las consultas.

Los tradicionales sistemas de RI tratan de igual manera las consultas y las preguntas. Es por ello que, en estos sistemas, la consulta "*¿Con qué grupo ha cantado Robbie Williams?*", del ejemplo 1.2, daría igual dársela en lenguaje natural que darle los términos "*grupo Robbie Williams*".

1.1.4. Indexación

A la hora de buscar términos en una colección de documentos de texto de un tamaño considerable, es necesario tratar la información previamente y estructurarla de cierta forma para que la búsqueda sea eficiente. Costaría mucho buscar un término de forma secuencial en una colección de documentos de varios gigabytes. A este proceso se le llama *indexar la colección de documentos.*

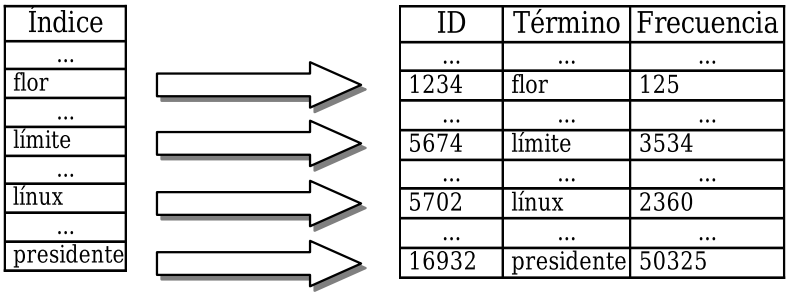


Figura 1.4: Ejemplo de un índice

Para el proceso de indexación se recorre la colección de documentos extrayendo los términos que vayan apareciendo. Al mismo tiempo se va construyendo una lista de documentos para cada término distinto. En dicha lista se relaciona el término con los documentos en los que aparece. En sistemas como el *mg* [WBM94] se debe, previamente, dividir los documentos en pasajes[1], al ser éste un sistema orientado a pasajes, y los términos ya no se relacionan con los documentos sino con los pasajes en los que aparecen.

Antes de construir un índice, se puede utilizar algún *stemmer* o *lematizador* para reducir el número de términos (mirar apartado 1.1.8).

Encontrar una palabra en un índice suele hacerse de una forma muy rápida al estar los índices representados con alguna estructura de datos que facilita su búsqueda, como árboles binarios, tries o tablas hash.

1.1.5. Ficheros invertidos

Los ficheros invertidos ([WBM94]) son un tipo especial de índices, en los cuales se relaciona cada término encontrado en la colección de documentos con todos los documentos (o pasajes) en los que dicho término aparece. Así, por cada término, se asocia una lista de longitud variable de identificadores de documentos (o pasajes) [BYRN99]. Éste es quizás el método más natural de indexación, pues corresponde muy estrechamente con el índice de un libro [WBM94].

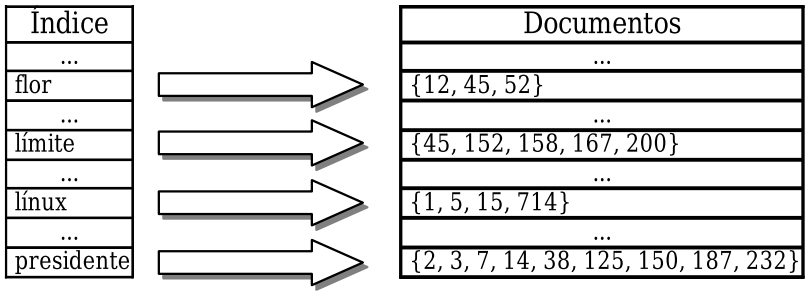


Figura 1.5: Ejemplo de un fichero invertido

[1]Para más información sobre pasajes mirar el apartado 1.1.9

Los términos de los ficheros invertidos están indexados para poder acceder a ellos de una forma eficiente. Así, para buscar los documentos en los que aparece un término, únicamente se buscan en el índice y se extrae la lista de documentos asociada a dicho término. En estas listas de documentos también pueden incluirse el peso específico del término para cada documento y otra información relevante, como el número de ocurrencias del término, etc. En una consulta más compleja, como por ejemplo una búsqueda booleana que requiera de los términos x_1 y x_2, la lista de documentos para la búsqueda x_1 *and* x_2 se podría obtener como la intersección de la lista de documentos de ambos términos; para las operaciones *or* se utilizaría la disyunción; y para las operaciones *not* se obtendría el complementario, es decir, todos los documentos que no estuvieran en la lista.

1.1.6. Palabras comunes o *stopwords*

A la hora de realizar una búsqueda por una lista de términos, existen un conjunto de palabras que no nos podrá decir nada respecto a si un documento es relevante o no. Esto es debido a que son palabras muy comunes que aparecen en casi todos los documentos. Los artículos, preposiciones, conjunciones, etc. son algunos ejemplos de dichas palabras comunes. Para ahorrar espacio en los ficheros invertidos y tiempo de búsqueda es muy común eliminar dichos términos de los ficheros invertidos ([WBM94]).

Hoy por hoy, hay dos mecanismos que funcionan bastante bien para detectar y eliminar dichas palabras. El mecanismo más frecuente es obtener una lista de palabras comunes creada a mano para cada idioma[2]. El número de palabras comunes puede variar bastante de un idioma a otro. Por ejemplo, para el inglés se han detectado unas 571 palabras mientras que para el español, únicamente, unas 189.

Pero la creación de estas listas requiere la intervención de un humano. La ley de Zipf (mirar sección 1.1.1) nos permite obtener una lista de palabras comunes de forma automática. Aunque el mejor límite para diferenciar palabras comunes de palabras relevantes se tiene que obtener mediante la experimentación.

[2]En la dirección Web http://www.unine.ch/info/clef/ se puede obtener las palabras comunes para 15 idiomas, entre ellos inglés, francés, español, árabe, etc.

a	se	de	del	un	una
unas	unos	uno	sobre	todo	también
tras	otro	algún	alguno	alguna	algunos
algunas	ser	es	soy	eres	somos
sois	estoy	está	estamos	estáis	están
como	en	para	atrás	porque	por
que	estado	estaba	ante	antes	siendo

Tabla 1.3: Ejemplo de palabras comunes o *stopwords* para el español

1.1.7. Términos relevantes o pivotes

Los términos relevantes o pivotes son aquellos que ayudan a localizar documentos o pasajes relevantes a una consulta dada por el usuario. Así, estos términos dependen en gran medida de la consulta realizada, de la colección de documentos y del sistema de RI utilizados.

Algunos términos pueden ser relevantes para una consulta y no serlo para otra. Un ejemplo muy sencillo y práctico es el término "*IRA*" que en español puede tener dos significados: el futuro del verbo '*ir*"[3] o las siglas del Ejército Republicano Irlandés. Así, y siguiendo con el ejemplo, esta palabra sería muy importante para la pregunta "*¿Cuándo se formó el IRA?*" pero podría tener muy poca importancia en la pregunta "*¿A qué país irá Jacques Santer en noviembre?*".

La colección de documentos también influye en la relevancia de los términos puesto que, si un término aparece muchas veces, tiene menos poder de discriminación que otros términos con muy pocas ocurrencias en la colección. Por ejemplo, si realizamos búsquedas en una colección de documentos que únicamente hablen de las noticias relacionadas con el presidente español *José Luis Rodríguez Zapatero*, sería inútil buscar por el término "*Zapatero*" puesto que esta búsqueda nos devolvería toda la colección de documentos y, por lo tanto, no nos habría ayudado a dis-

[3] Aunque la forma verbal que representa el futuro de "*ir*" se escribe en minúsculas y con acento en la última letra ("*irá*"), es muy común, a la hora de indexar las colecciones de documentos, eliminar los acentos y almacenar todas las palabras en minúsculas, por lo tanto, estos dos términos se suelen confundir.

criminar o a localizar información relevante. También puede ocurrir que existan términos en la colección que aparezcan en muy pocos documentos pero que tampoco sean términos relevantes. Este es el caso de los errores tipográficos.

Los términos relevantes también dependen del sistema de RI utilizado. Hay sistemas que no tienen en cuenta las palabras comunes y otros sistemas que las utilizan para afinar mejor la búsqueda y, por lo tanto, estos términos adquieren cierta relevancia. Otros sistemas no catalogan los términos en relevantes o no relevantes sino que, a cada término, se le asigna cierto grado de relevancia utilizando, para ello, diferentes medidas.

1.1.8. *Stemming* y lematización

El *stemmer* se utiliza para truncar las palabras de tal forma que nos quedemos con la raíces. Mientras que el *lematizador* se queda con el lema de la palabra.

El uso del stemming o de la lematización reduce el tamaño de los ficheros invertidos y aumenta la cobertura en las búsquedas. Esto es debido a que muchas palabras distintas, pero que tengan el mismo lema o la misma raíz, se tratarán como si fueran el mismo término a la hora de indexar. Así, si por ejemplo buscamos el término *bombardeando*, no sólo encontraremos los documentos (o pasajes) en los que aparezca dicho término, sino además aquellos documentos (o pasajes) en los que aparezcan *bombardear*, *bombardeado*, *bombardearon*, etc. Aunque el asociar términos sólo por tener la misma raíz o lema puede reducir la precisión al confundir términos que, de otra forma, no estarían relacionados.

Término	**Lematización**	**Stemming**
Presidente	Presidente	President
Límite	Limitar	Limit
México	México	México
flores	flor	flor

Tabla 1.4: Diferencias entre lematización y stemming

1.1.9. Pasaje

Un *pasaje* es un trozo de texto que pertenece a un documento (figura 1.6). Es muy común dividir el documento en pasajes para obtener mejores resultados en la búsqueda ([LP01]). Los resultados mejoran debido, principalmente, a dos razones: (i) es complicado comparar una consulta de unos pocos términos con documentos que pueden tener decenas o cientos de términos; y (ii) los documentos de tamaño considerable tienen ventaja sobre los cortos ya que poseen más términos y, en consecuencia, es más probable que incluyan los términos de la consulta. Además, en los documentos grandes un término dado tiene mayor verosimilitud de repetirse.

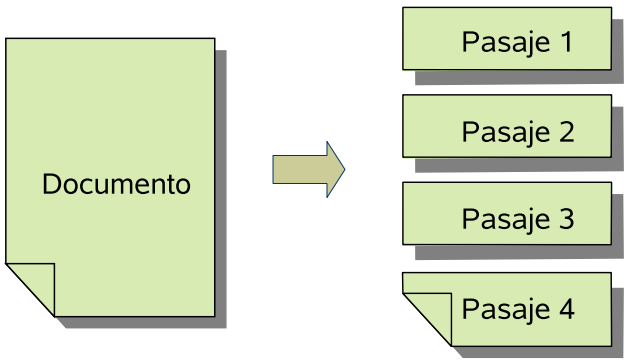


Figura 1.6: Pasajes

Muchos modelos, entre ellos el modelo Espacio Vectorial ([Sal71]), intentan compensar el tamaño de estos documentos multiplicando su peso por un factor que decrece con respecto al número de términos. Otros sistemas de RI dividen los documentos en pasajes para equiparar su tamaño entre si y entre la consulta y, de esta forma, poder compararlos mejor. Esto solventa los dos problemas mencionados anteriormente: equilibra los tamaños de los documentos y, además, los reduce a un tamaño equiparable al de la consulta. Es más, usar pasajes en vez de documentos completos permite encontrar trozos de texto relevantes en un documento, en general, poco relevante.

También, gracias a la división por pasajes, se puede discernir aquellos documentos que contienen todos o la mayor parte de los términos

de la consulta muy esparcidos o, por el contrario, más aglutinados. En los documentos donde los términos de la consulta están más aglutinados tienden a ser más relevantes. Un sistema de RI basado en pasajes colocaría en las primeras posiciones del *ranking* a estos documentos pues contendrían pasajes con un número de términos mayor. Sin embargo, en los documentos en los que los términos relevantes estuvieran esparcidos sería más improbable encontrar pasajes que tuvieran muchos términos relevantes.

1.1.10. Expansión de la pregunta

Cuando se hace una consulta a un sistema de RI, el sistema devuelve aquellos documentos en los que aparecen los términos de la consulta, descartando otros documentos relevantes al no utilizar en ellos exactamente los mismos términos. Para evitar este problema se recurrió a la expansión de la pregunta utilizando alguno de estos dos métodos:

Thesaurus o base de datos de conocimiento

Los *thesaurus* o base de datos de conocimiento son diccionarios extendidos que incorporan a cada palabra información extra. Al contrario que los diccionarios clásicos, los thesaurus tienen la propiedad de que son capaces de relacionar diferentes entradas del diccionario entre sí utilizando una o más relaciones.

Se utilizan estas bases de datos de conocimiento para añadir nuevos términos a la consulta que tienen algún tipo de relación sintáctica o semántica con los términos originales. Es muy común, en estos sistemas, añadir los sinónimos de los términos.

En la figura 1.7 aparece un sencillo ejemplo en el que un usuario realiza la consulta "*capital croacia*". El sistema de RI que utiliza un thesaurus añade a la consulta los términos "*república*", "*mayúscula*" y "*letra*" que son palabras relacionadas de alguna forma con los términos de la consulta.

Uno de los thesaurus más utilizados en este tipo de expansión es Wordnet ([MBF+90]). Wordnet es una base de datos formada por relaciones semánticas entre las palabras, las cuales están agrupadas por sus significados. La relación central que utiliza Wordnet para estructurar su información es la sinonimia, aunque también expresa otras relaciones

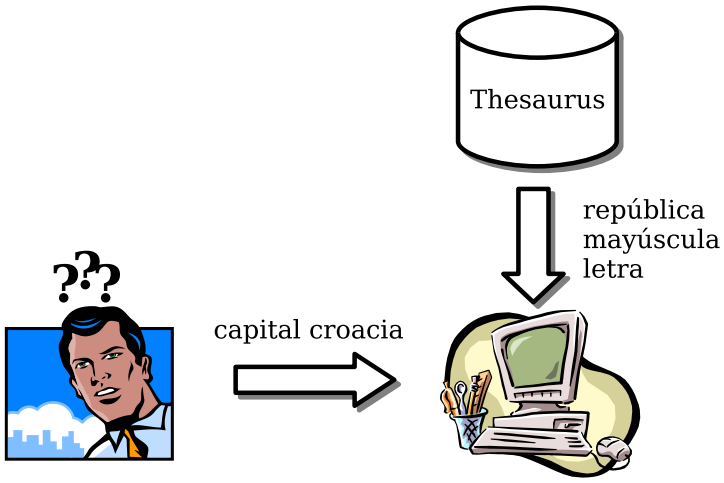


Figura 1.7: Ejemplo de expansión de la pregunta a partir de un thesaurus

como meronimia, homonimia, hiperonimia, hiponimia, antonimia, etc.

Los modelos basados en thesaurus no han logrado mejorar los resultados en tareas de RI debido, sobre todo, a que añaden ruido a la consulta, es decir, términos no relacionados con el tema de la consulta, considerando relevantes documentos que no lo son. Esto es debido a que cada palabra puede tener relaciones con diferentes términos en función del sentido que tenga en un contexto determinado. La mejora que pueden aportar estos diccionarios es escasa, incluso aunque la búsqueda de relaciones entre palabras sea supervisada por humanos ([Voo94, LP01]).

Realimentación

La realimentación (también conocida como *relevance feedback*) es la estrategia de realimentación más popular ([BYRN99]). Se trata de un método heurístico para la expansión de la consulta que consiste en añadir términos relacionados con los términos de la consulta. Experimentos usando el sistema de RI Smart ([Sal71]) y después experimentos usando un modelo probabilístico ([RSJ88]) muestran el buen rendimiento en precisión de estos sistemas cuando se utiliza la realimentación.

Para realizar la realimentación existen dos estrategias muy extendidas:

Análisis local En este método se realiza una primera consulta y se obtiene los n primeros documentos. De estos n documentos se extraen los términos más frecuentes y se vuelve a lanzar otra consulta, pero añadiendo a la original los nuevos términos. Este método supone que los documentos relevantes son los n primeros recuperados de la búsqueda y, por lo tanto, incluirán otros términos relevantes que se repetirán con frecuencia ([JJJW99, XC96]).

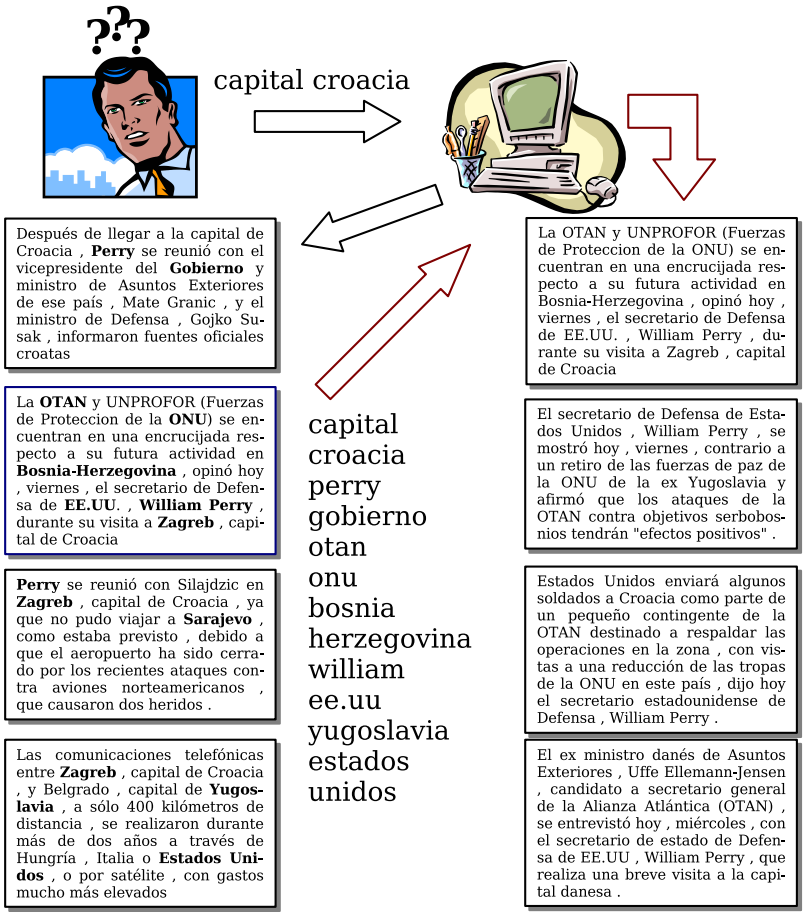


Figura 1.8: Ejemplo de realimentación

Análisis global La diferencia crucial entre el análisis local y global es que el primero realiza una búsqueda en un subconjunto de los pasajes recuperados (figura 1.8), mientras que el análisis global realiza, durante el periodo de indexación, una matriz de coocurrencias ([QF93]). Así, durante la búsqueda, añade automáticamente aquellos términos que estén fuertemente relacionados con los términos de la consulta en esta matriz de coocurrencias.

Los trabajos desarrollados por Donna Harman suponen un buen estudio comparativo de las diferentes técnicas de realimentación ([Har88, Har92a, Har92b]).

1.2. Áreas de la Recuperación de Información

Los sistemas de RI iniciaron su andadura como motores de búsqueda que encontraban documentos relacionados, de alguna forma, con unas pocas palabras claves. Estos documentos pertenecían a colecciones donde los textos no tenían ninguna estructura preestablecida. A medida que aumentaba la tecnología y hacía posible aplicar nuevos algoritmos de RI, se empezó a crear nuevas líneas de estudio. Actualmente, existen innumerables ramas de investigación pero todas se pueden aglutinar en tres principales áreas: la RI *Ad hoc*, la Extracción de la Información (EI) y la Búsqueda de Respuestas (BR). Las diferencias entre estas áreas radican, principalmente, en el enfoque que se da al objetivo final que se desea conseguir ([LP01]). Dicho objetivo se determina por las necesidades de información del usuario. Así, si el usuario necesita como respuesta una lista de documentos relacionados con la consulta, si necesita un respuesta concreta a una pregunta, o si necesita, de alguna manera, extraer información relevante según un modelo o formulario, entonces se utilizará un sistema u otro.

En los siguientes apartados se analiza con más detalle cada una de éstas líneas de investigación.

1.2.1. Recuperación de la Información *Ad hoc*

La RI *Ad hoc* es la principal área de investigación, la más antigua y en la que se han invertido los mayores esfuerzos (figura 1.9). Muchos sistemas comerciales de gran éxito están basados en sistemas de RI Ad hoc que recuperan documentos a través de Internet. Entre las compañías más destacadas que utilizan estas tecnologías podemos encontrar, por ejemplo, a Google[4], Yahoo[5] o MSN[6]. Estos sistemas permiten que cualquier usuario realice consultas y, después, buscan páginas Web (documentos) relevantes a las consultas utilizando diferentes heurísticas. Para estos sistemas la parte pública y accesible de Internet sería su inmensa colección de documentos.

Figura 1.9: Sistemas de Recuperación de Documentos

Básicamente la RI Ad hoc consiste en recuperar aquellos documentos relevantes de una colección a partir de una consulta ([Sme97]). Estos sistemas deben devolver una lista ordenada descendentemente por relevancia llamada *ranking* ([Sal89]). De esta forma, a cada documento se le asocia un valor que mide la similitud o relevancia con respecto a la

[4] www.google.es
[5] www.yahoo.es
[6] search.msn.com

consulta. El valor de esta similitud depende del modelo escogido y de las funciones de pesado de los términos y de los documentos. Aunque, en un principio, la consulta puede estar compuesta por cualquier cantidad de texto (desde un documento, un resumen, un frase en lenguaje natural, etc.), lo normal es que las consultas a estos sistemas se efectúe con unos pocos términos relevantes.

Existen multitud de técnicas para que este proceso se haga eficientemente. Los ficheros invertidos son una de las herramientas más poderosas para agilizar las búsquedas. Tratan de asociar cada término con los documentos en los que aparece ([BYRN99]). Además, aunque la formulación formal de muchas de las técnicas utilizadas para la RI Ad hoc aparece una comparación de cada término de cada documento con cada término de la consulta, en la práctica, y gracias a los ficheros invertidos, se comparan únicamente aquellos documentos en los que aparece algún término de la consulta. Es más, ni siquiera se comprueban todos los términos de los documentos obtenidos, sino sólo aquellos que también aparecen en la consulta, puesto que la contribución del resto de términos del documento suele ser nula.

Antes de realizar cualquier búsqueda en estos sistemas de RI Ad hoc, se deben preparar los datos para agilizar las búsquedas. A este proceso se le llama *indexar la colección de documentos*. Así, cuando se realiza la búsqueda, ésta utiliza los recursos y datos generados durante la indexación de la colección (figura 1.10).

Durante la indexación (que normalmente se realiza una única vez), los documentos son analizados y su contenido se almacena en estructuras de datos para que la búsqueda sea eficiente. Es muy común eliminar acentos, limpiar los términos de caracteres especiales, realizar procesos de stemmer o lematización y eliminar las palabras comunes para reducir el corpus y aumentar la cobertura. Además, algunos sistemas pueden realizar la división de los documentos durante la etapa de indexación para trabajar con fragmentos de texto más pequeños y homogéneos. También se puede, en esta etapa, precalcular los pesos de los documentos y términos, aunque esto dependerá, en gran medida, del modelo utilizado.

Una vez indexada la colección de documentos y obtenido los diferentes índices y ficheros invertidos, el sistema ya está preparado para realizar consultas. De esta forma, se aprovechan las estructuras de datos creadas durante el proceso de indexación para recuperar, en muy poco tiempo,

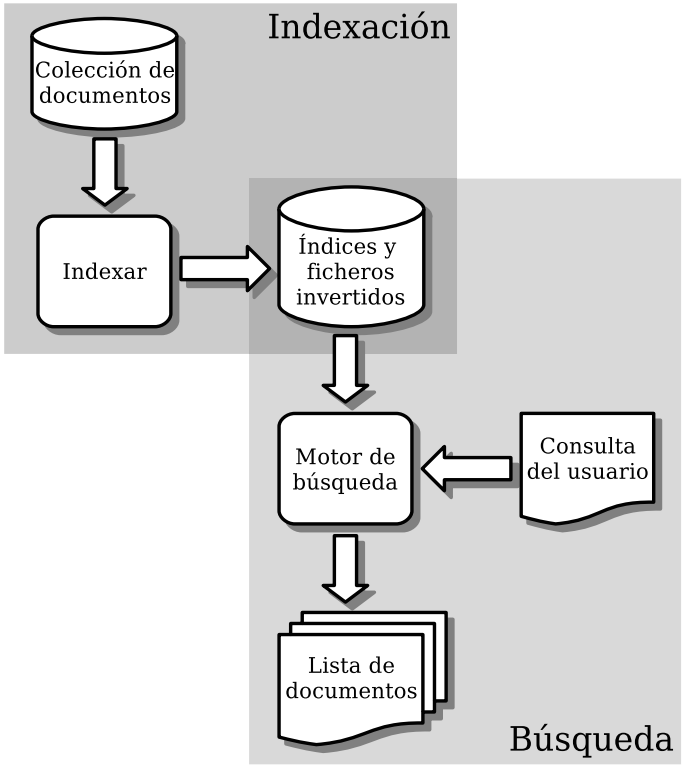


Figura 1.10: Indexación y búsqueda

lo que tardaría horas en resolverse si los documentos no estuvieran procesados.

En la figura 1.10 se puede observar las dos etapas y cómo estas tienen un punto en común en los índices y ficheros invertidos. Así, todos los procesos realizados a los términos de los documentos (stemmer, lematización, eliminación de caracteres extraños, etc.) durante la tarea de indexación deben ser repetidos durante la búsqueda sobre los términos de la consulta. Si no fuera así, podríamos buscar una palabra acentuada sobre un corpus que no incorporase los acentos y, en consecuencia, el sistema no sería capaz de encontrar dicha palabra.

1.2.2. Búsqueda de Respuestas

La BR tiene como objetivo devolver la respuesta exacta a una pregunta del usuario realizada en lenguaje natural (figura 1.11). Para ello, el primer paso es seleccionar aquellos trozos de texto de nuestra colección de documentos en donde se crea que pueda estar la respuesta. Este primer paso es típico que lo realice un sistema de RI Ad hoc que devuelva pasajes en vez de documentos completos.

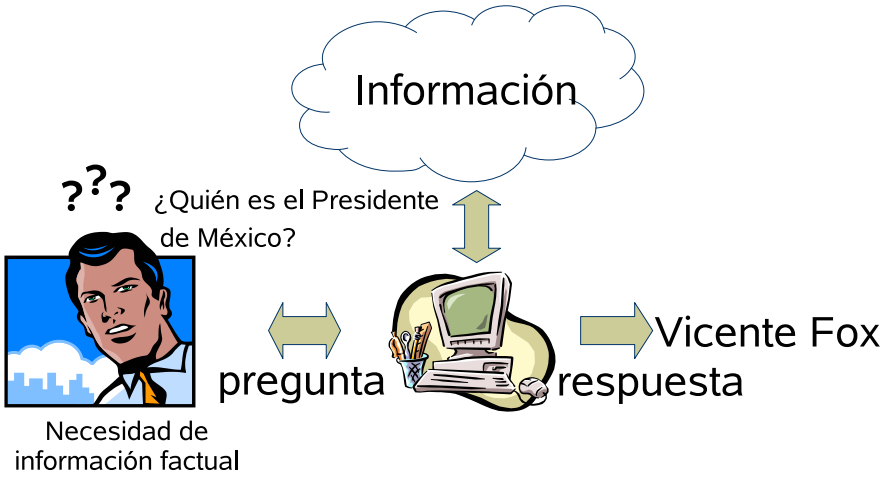


Figura 1.11: Sistemas de Búsqueda de Respuestas

Una vez obtenido estos pasajes se utilizan diferentes técnicas de Procesamiento del Lenguaje Natural (PLN) o estocásticas para obtener la respuesta. En este paso es cada vez más evidente el uso de la Web, en concreto su redundancia, para obtener o validar las respuestas ([CLMS+00, BLB+01, Buc01, BD01, DYC03, RFQ+02]).

1.2.3. Extracción de la Información

Un área de la RI donde el PLN juega un papel crucial es en la Extracción de la Información (EI).

El objetivo de la EI es rellenar una plantilla o formulario a partir de la información de la colección de documentos. La información de dicha

plantilla obtenida se usará para una determinada tarea muy específica. Por ejemplo, se puede desear obtener el nombre, DNI y dirección de las personas contratantes que aparecen en documentos notariales (figura 1.12).

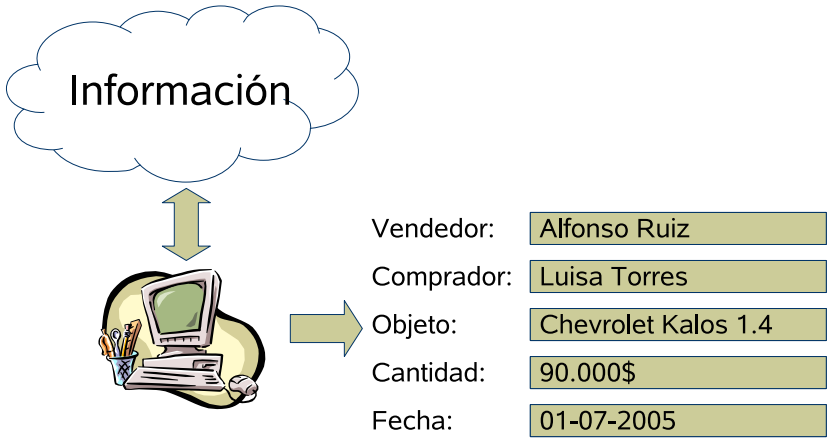


Figura 1.12: Extracción de la Información

También es común, en algunos sistemas de BR, el uso de técnicas de EI para rellenar plantillas con nombre de entidades que encajen con el tipo de la respuesta esperado.

1.3. Motivación y objetivos de la tesis

Como veremos en el capítulo 2, los sistemas de RI utilizados en BR no consiguen devolver un porcentaje muy alto de pasajes con la respuesta correcta a partir de una pregunta en lenguaje natural. Así, nuestro principal objetivo es conseguir un sistema de RI que mejore el rendimiento en tareas de BR multilingües. Al ser una tarea multilingüe, es decir, que pueden intervenir varios idiomas europeos, nuestro sistema debe ser tan independiente como sea posible del lenguaje. Además, debe estar adaptado para buscar los pasajes que tengan la respuesta, en vez de documentos relevantes, a partir de una pregunta realizada por el usuario en lenguaje natural, en vez de con unos pocos términos.

Entre otros objetivos nos proponemos:

- Realizar un nuevo sistema de Recuperación de Pasajes (RP) orientado a BR.

  Como se explicará en el capítulo 2, la mayoría de los sistemas de BR, o bien utilizan clásicos sistemas de RI que no están orientados a la BR, o bien son muy dependientes del idioma. Los primeros devuelven un alto número de documentos que no contienen la respuesta a la pregunta del usuario y, por lo tanto, el sistema de BR no puede responder a dichas preguntas; los segundos son sistemas muy difíciles de adaptar a nuevos idiomas y, además, no obtienen una sustancial mejora que justifique esta alta dependencia. Nosotros nos hemos propuesto desarrollar un sistema de RI cuyo objetivo es obtener pasajes con mayor probabilidad de contener la respuesta, en vez de pasajes relevantes a la consulta.

  Los actuales sistemas de RI intentan mejorar tanto la cobertura como la precisión de pasajes o documentos relevantes. A nuestro sistema le interesará más la cobertura de la respuesta. Es decir, el número de preguntas cuyas respuestas se pueden contestar observando los pasajes devueltos por el sistema. Nos interesa un sistema de RI que obtenga en los primeros puestos, pasajes con la respuesta correcta, aunque la lista de pasajes devueltos no incluya todos los pasajes del sistema con la respuesta, ni otros pasajes relacionados. Para ello nos basaremos, principalmente, en la redundancia de las colecciones de documentos y/o Web. En otras palabras, a partir de una colección de documentos suficientemente extensa como para que la respuesta se pueda encontrar escrita de varias formas, nos proponemos encontrar aquel pasaje con mayor verosimilitud de contener la respuesta y que, además, sea más fácil de extraer.

  Con el fin de alcanzar este objetivo, nos proponemos encontrar la respuesta que esté contenida en una expresión lo más parecida posible a la pregunta para facilitar después su extracción. Entre todas las formas de expresar la respuesta, nos centraremos en obtener en las primeras posiciones del ranking aquellas que se parezcan más a la pregunta del usuario. Esto es posible gracias a la alta redundancia de las colecciones de documentos y/o Web, que expresan muchas veces la respuesta de muchas formas distintas.

- El sistema se basará en la búsqueda de estructuras y utilizará di-

ferentes modelos de pesado y cálculo de similitud entre los pasajes y la pregunta.

Para obtener pasajes con mayor probabilidad de incluir la respuesta en vez de, simplemente, pasajes relevantes, nos proponemos intentar encontrar pasajes en que la estructura de la pregunta esté presente en el pasaje, o si no, la estructura de mayor peso posible. Para llevar a cabo esta tarea, es necesario desarrollar un mecanismo eficiente que busque estructuras en vez de palabras claves. El principal problema de la búsqueda de estructuras es encontrar una forma rentable (tanto en coste espacial como temporal) de encontrar las estructuras. La razón de que sea complicada la búsqueda de estructuras es debido al hecho de indexar la colección de documentos antes de conocer las preguntas o, más concretamente, el tamaño de éstas. En la presente tesis veremos cómo hemos conseguido salvar este inconveniente utilizando una aproximación heurística al problema.

- Implementar y adaptar sistemas de RI tradicionales.

  Una vez obtenidos nuestros modelos de RI, es interesante compararlos con los sistemas de RI más utilizados en la BR multilingüe. Por lo tanto, el primer paso será desarrollar y adaptar los sistemas de RI clásicos utilizados en la bibliografía para las tareas de BR. Pero no es sólo el único objetivo. Como veremos en el capítulo 4, nuestros modelos se basarán en un motor de búsqueda tradicional. Podremos usar estos sistemas clásicos para poder conseguir un motor de búsqueda que se integre correctamente como primera etapa de nuestro sistema.

  También pensamos implementar un nuevo motor de búsqueda especialmente adaptado a las necesidades de nuestros modelos. Debido a las especiales características de nuestro sistema de RI, puede que sea necesario un motor de búsqueda especializado que entregue pasajes más idóneos como entrada al sistema. Así, podremos comprobar nuestras suposiciones sobre qué tipo o qué características deben tener los pasajes para que nuestros modelos de búsqueda de estructuras funcionen mejor.

- El sistema debe ser tan independiente del idioma como sea posible.

En tareas multilingües en los que la pregunta y la colección de documentos está en varios lenguajes, o para tareas Cross-Language en los que la pregunta está en un idioma y los documentos en otro, o, simplemente, en los sistemas de BR que deben trabajar con varios idiomas, es necesario utilizar sistemas que se adapten bien a diferentes lenguajes sin demasiados cambios y utilizando recursos que sean fácilmente accesibles. Para ello es necesario que el sistema de RP sea tan independiente como sea posible del habla y que se pueda adaptar fácilmente a otros idiomas.

Veremos, en el capítulo 5, que nuestro sistema no sólo es altamente independiente del idioma, sino que mejora a otros sistemas que no lo son, tanto en precisión como en cobertura y redundancia de la respuesta.

- Realizar pruebas con diferentes idiomas como pueda ser el español, inglés, francés e italiano.

  Por supuesto, necesitaremos valorar la independencia del lenguaje aplicándo el sistema a diferentes idiomas y observar si los resultados son similares, independientemente del idioma utilizado. Cabe destacar que, por el momento, el sistema sólo será aplicable a idiomas no aglutinativos, puesto que se basa en la separación léxica para indexar los términos.

1.4. Organización de la tesis

El resto de la presente tesis se estructura de la siguiente forma: en el capítulo 2 estudiaremos los sistemas actuales de RI, desde una visión global de los sistemas clásicos de RI (sección 2.1), hasta una visión ya más específica de los sistemas de RP (sección 2.2), pasando, por supuesto, por los sistemas de BR (sección 2.4), ya que nuestro sistema de RP es un sistema de RI que está orientado a la BR; a continuación, en el mismo capítulo, describiremos los dos congresos más importante en RI y BR en idiomas europeos y, en cuyos datos, se basa nuestra experimentación y resultados; seguidamente explicaremos nuestro sistema y los modelos desarrollados (capítulo 4); durante todo el capítulo 5, mostraremos los corpora y sistemas utilizados, los distintos experimentos evaluados y los

resultados obtenidos; y, finalmente, en el capítulo 6, esbozaremos las conclusiones más relevantes y los trabajos futuros.

Además, se ha incluido una serie de apéndices que pueden ser muy interesantes al lector o investigador. En el apéndice A se describen todas las abreviatura utilizadas en la presente tesis, para que el lector pueda utilizarlo en el caso de querer leer capítulos sueltos. El apéndice B está destinado a los usuarios que quieran utilizar JIRS como sistema de RI. En este apéndice se explican los pasos básicos para su uso.

Capítulo 2
Estado del arte

2.1. Sistemas de Recuperación de Información

Los sistemas de RI han sido, tradicionalmente, sistemas de *Recuperación de Documentos* (RD) (figura 1.9). Estos sistemas intentan encontrar, a partir de una consulta de términos realizada por el usuario, aquellos documentos relevantes a dicha consulta.

Los sistemas de RD son los sistemas de RI más utilizados hoy en día. Principalmente, aunque con pequeñas variaciones, existen únicamente tres grandes modelos ([BYRN99]): el modelo *Booleano*, el modelo *Espacio Vectorial* (EV) y el modelo *Probabilístico*.

2.1.1. Modelo booleano

El modelo Booleano se basa en realizar operaciones booleanas de pertenencia sobre los términos de la consulta.

En la figura 2.1 se puede ver un ejemplo de este modelo. En el cual, el usuario ha realizado la consulta "*RI modelo booleano*" y el sistema obtiene diferentes conjuntos de documentos donde uno o varios de los términos de la consulta aparecen. Dependiendo de las necesidades de la consulta nos puede interesar obtener aquellos documentos en los que aparecen los tres términos, aparezca uno y los otros no o, en general, cualquier operación booleana con los conjuntos.

El modelo Booleano tiene el inconveniente que determina de forma booleana si un documento pertenece o no a un conjunto, es decir, sólo se

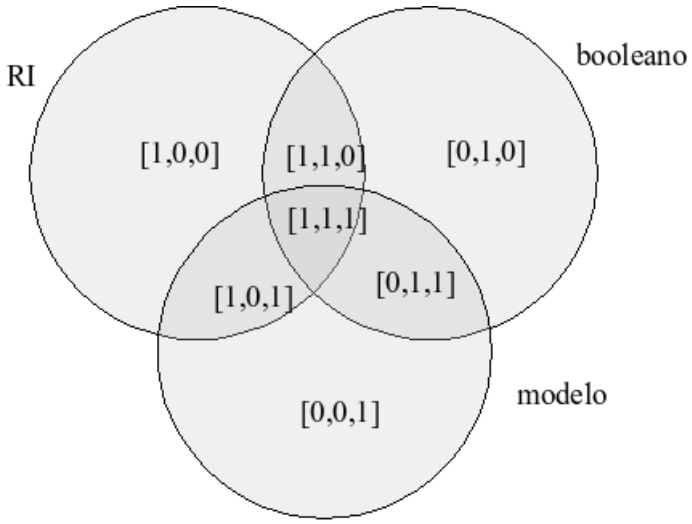


Figura 2.1: Modelo Booleano

puede saber si los documentos son relevantes o no, pero no qué grado de relevancia tienen. En la tabla 2.1 se puede observar los pesos asignados a diferentes documentos que contienen el término A, el término B o ambos, tanto para las consultas con el operador **and** como con el operador **or**.

	Términos		**Similitud con la consulta**	
	A	**B**	**(A or B)**	**(A and B)**
Documento D_1	1	1	1	1
Documento D_2	1	0	1	0
Documento D_3	0	1	1	0
Documento D_4	0	0	0	0

Tabla 2.1: Peso de los términos A y B con el modelo Booleano para las consultas *and* y *or*

,

Estos modelos tienden a devolver demasiados documentos o ninguno ([Har92c]), además no son capaces de tratar mejor aquellos documentos que contienen todos los términos de la consulta que otros con sólo un

término ([SB87]). Otros modelos, como el modelo *Booleano Extendido* (BE) ([SFW83]) o el modelo *Booleano Difuso* ([Zad65]), fueron creados para suplir estas deficiencias.

2.1.2. Modelo Booleano Difuso

El modelo Booleano Difuso está basado en la teoría de conjuntos difusos (*fuzzy set theory*) ([Boo80, Boo81, WK79, BK81]). En el modelo estándar de conjuntos difusos de RI, los términos de los documentos tienen un peso y las consultas son procesadas como formulaciones booleanas. Por ejemplo, imaginemos que tenemos las consultas "A **or** B", "A **and** B" y "**not** A", un documento X con pesos $d_A(X)$ y $d_B(X)$ para los términos A y B, respectivamente, recibe los siguientes valores en un sistema de recuperación de conjuntos difusos:

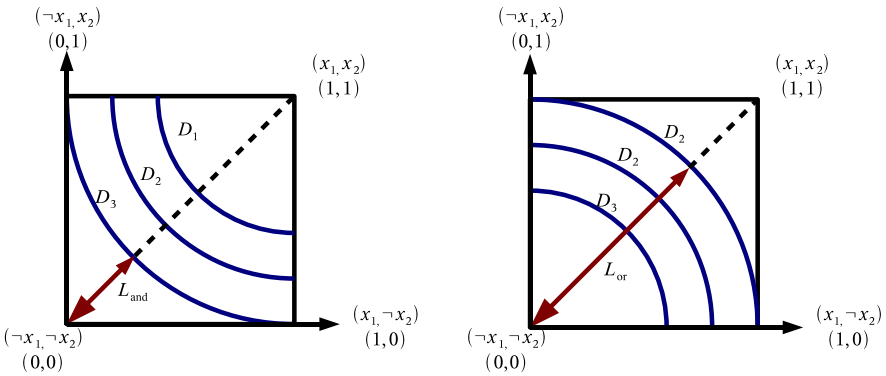
$$\begin{array}{l} max[d_A(X), d_B(X)] \text{ para la consulta } (A \textbf{ or } B) \\ min[d_A(X), d_B(X)] \text{ para la consulta } (A \textbf{ and } B) \\ 1 - d_A(X) \text{ para la consulta } (\textbf{not } B) \end{array} \tag{2.1}$$

Si los pesos de los términos asociados a los documentos se restringen a 0 y 1, entonces el modelo de conjuntos difusos es compatible con el modelo Booleano. Desafortunadamente, el sistema de RI basado en conjuntos difusos sufre una carencia de discriminación, ya que el valor de similitud del documento depende sólo del peso más alto o mínimo de los términos para las consultas **or** y **and** respectivamente. Además, es muy difícil adaptar el modelo de conjuntos difusos a situaciones donde el peso de los términos también depende de la consulta.

2.1.3. Modelo Booleano Extendido

Los modelos BE fueron introducidos por Salton para poder suplir los inconvenientes del modelo Booleano Difuso como fue descrito en [SFW83]. Los modelos BE emplean *operadores booleanos extendidos* (también llamados *soft boolean operators*). En el clásico modelo Booleano, los operadores devuelven únicamente dos valores: verdadero o falso. Pero los operadores booleanos extendidos pueden devolver un rango de valores entre 0 y 1 que dependerá del peso de los términos de la consulta y de cada documento.

Cuando sólo se tienen en cuenta dos términos, la asignación del valor de similitud de la consulta puede ser representada por un mapa bidimensional, como aparece en la figura 2.2, donde cada término es asignado a una coordenada distinta.



(a) x_1 and x_2 (b) x_1 or x_2

Figura 2.2: Ejemplo del modelo BE para los operadores *and* y *or*

Para las consultas del tipo **and** la situación **más** deseable es el punto del mapa (1, 1) que representa la situación dónde ambos términos aparecen; para las preguntas del tipo **or** la posición **menos** deseable es aquella donde ambos términos no aparecen, el punto (0, 0). Esto sugiere que un sistema de RI basado en el modelo BE debe ordenar los documentos en orden creciente a medida que incrementamos la distancia desde (1, 1) para las consultas **and** y en orden decreciente para las consultas **or** a medida que aumenta la distancia desde (0, 0). Para los documentos con los pesos d_A y d_B para los términos A y B, respectivamente, se puede calcular la distancia euclidiana con respecto al punto (0, 0) como $\sqrt{(d_A - 0)^2 + (d_B - 0)^2}$ y con respecto al punto (1, 1) como $\sqrt{(1 - d_A)^2 + (1 - d_B)^2}$. Para trabajar con la distancia normalizada es muy común dividir por la distancia máxima entre los puntos (0, 0) y (1, 1) de $\sqrt{2}$. Para términos cuyos pesos estén entre 0 y 1, Salton sugiere las ecuaciones 2.2 y 2.3 para las consultas **and** y **or** respectivamente.

$$sim(D, Q_{(A \textbf{ and } B)}) = 1 - \sqrt{\frac{(1 - d_A)^2 + (1 - d_B)^2}{2}} \qquad (2.2)$$

$$sim(D, Q_{(A \textbf{ or } B)}) = \sqrt{\frac{d_A^2 + d_B^2}{2}} \tag{2.3}$$

La tabla 2.2 visualiza los valores de similitud usando las fórmulas 2.2 y 2.3 cuando los pesos de los términos A y B valen 1 para las consultas *and* y *or*.

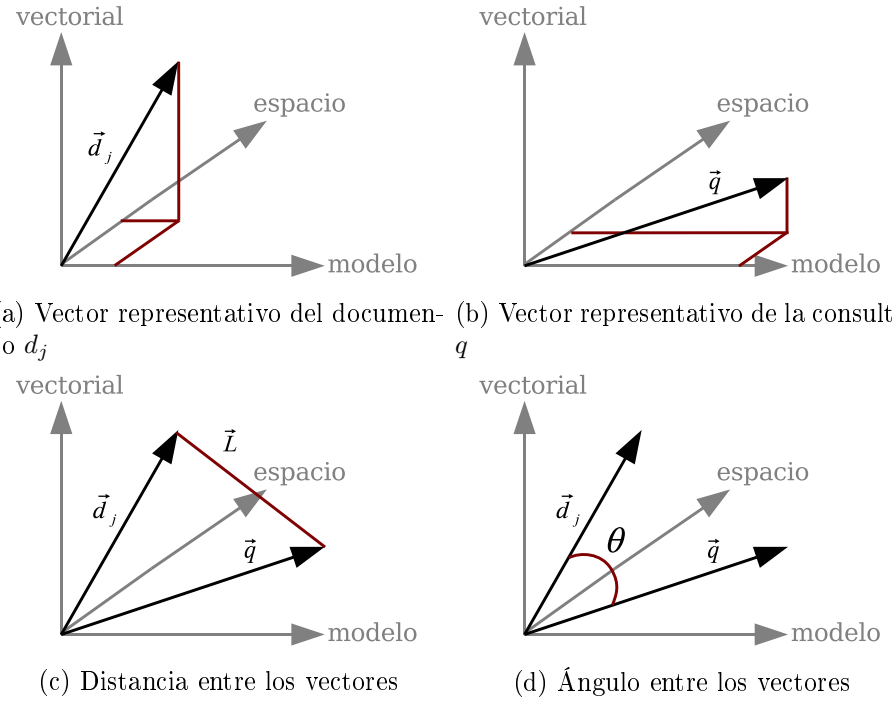
	Términos		**Similitud con la consulta**	
	A	**B**	**(A or B)**	**(A and B)**
Documento D_1	1	1	1	1
Documento D_2	1	0	$\frac{1}{\sqrt{2}}$	$1 - \frac{1}{\sqrt{2}}$
Documento D_3	0	1	$\frac{1}{\sqrt{2}}$	$1 - \frac{1}{\sqrt{2}}$
Documento D_4	0	0	0	0

Tabla 2.2: Peso de los términos A y B con el modelo BE para las consultas *and* y *or*

2.1.4. Modelo Espacio Vectorial

El modelo Espacio Vectorial (EV) trata de representar, en un espacio multidimensional, dos documentos mediante vectores teniendo en cuenta, típicamente, los términos del documento y su frecuencia. Una vez representado los documentos como vectores en un espacio multidimensional se calcula una medida de similitud entre ellos.

En el ejemplo de la figura se muestran dos vectores que representan dos documentos formados, únicamente, por tres términos: "*vectorial*", "*space*" y "*model*". Cada documento está compuesto por un número dado de cada uno de estos términos. Así, el vector $\vec{d_j}$ representará al documento d_j, como aparece en la figura 2.3a, mientras que el vector $\vec{q}$ de la figura 2.3b representa al documento q (que en este caso bien podría ser la consulta puesto que el modelo EV no hace distinción entre consulta y documento). Si ambos documentos tuvieran los mismos términos con igual peso serían representados por vectores exactamente iguales. En el ejemplo podemos ver que los términos "*vectorial*" y "*space*" se repiten más (o tienen más peso) en el documento d_j que en q y que el término "*model*" tiene mayor peso en el documento q.



(a) Vector representativo del documento d_j

(b) Vector representativo de la consulta q

(c) Distancia entre los vectores

(d) Ángulo entre los vectores

Figura 2.3: Ejemplo del modelo EV

Con los vectores $\vec{d_j}$ y $\vec{q}$ que representan a los documentos d_j y q respectivamente, podemos realizar una comparación para calcular su similitud. Para este cálculo podríamos medir la distancia entre los vectores como aparece en la figura 2.3c, pero esta función de similitud tiene el inconveniente de que para documentos de tamaños dispares la similitud cambia considerablemente, independientemente de que contengan la misma proporción de términos. Por esta razón no es eficaz cuando se compara una consulta de unos pocos términos con un documento que tiene decenas o cientos.

La función de similitud más conocida y más utilizada en el modelo EV es la función del coseno (ecuación 2.4) que no mide la distancia entre los vectores sino el ángulo que lo forman (o, más concretamente, el coseno del ángulo).

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{j,q}^2}} \tag{2.4}$$

Donde $w_{i,j}$ y $w_{i,q}$ representan los pesos asociados del término i-ésimo en el documento d_j y la consulta q respectivamente y t es el número total de términos del sistema.

Así, un documento d_j y una consulta q se representan en un espacio t-dimensional de vectores como se puede apreciar en la figura 2.3d. El modelo de EV propone evaluar el grado de similitud entre el documento d_j con respecto a la consulta q como la correlación entre los vectores $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j}$ y $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{t,q})$. Usando el coseno del ángulo entre estos dos vectores tendríamos la ecuación 2.4.

Aunque existen muchas fórmulas de pesado de términos, la más conocida para el modelo EV es la función *term-frequency/inverse document frequency* (tf/idf) ([BYRN99]). Esta función se muestra en la ecuación 2.5.

$$w_{i,j} = \frac{freq_{i,j}}{max_l freq_{l,j}} \times log\frac{N}{n_i} \tag{2.5}$$

Donde N el número total de documentos en la colección; n_i el número de documentos donde el término i-ésimo aparece; $freq_{i,j}$ y $freq_{l,j}$ son las frecuencias de los término i y l-ésimo en el documento j. Así el tf/idf calcula la frecuencia relativa del término i-ésimo con respecto al documento j y lo multiplica por la proporción de documentos en los que aparece ese término en la colección.

Para calcular el peso de los términos de la consulta [SB87] sugieren:

$$w_{i.q} = \left(0{,}5 + \frac{0{,}5 \cdot freq_{i,q}}{max_l \cdot freq_{l,q}}\right) \times log\frac{N}{n_i} \tag{2.6}$$

Donde $freq_{i,q}$ es la frecuencia del término i-ésimo en la consulta q.

2.2. Sistemas de Recuperación de Pasajes

Utilizando los clásicos modelos de RI mencionados en el apartado anterior, es complicado comparar una simple consulta de unos pocos

términos con un documento completo. Si no se realiza ningún tipo de normalización, documentos muy extensos pueden tener preferencia sobre los documentos más cortos puesto que los primeros tienen más términos y más repeticiones pese a que puedan ser menos relevantes. Es más, aunque un documento sea, en conjunto, poco relevante, puede contener un párrafo o trozo de texto tremendamente relevante.

Los sistemas de Recuperación de Pasajes (RP) son sistemas que obtienen, en vez de documentos completos, fragmentos de documentos que son relevantes a la consulta del usuario (figura 2.4).

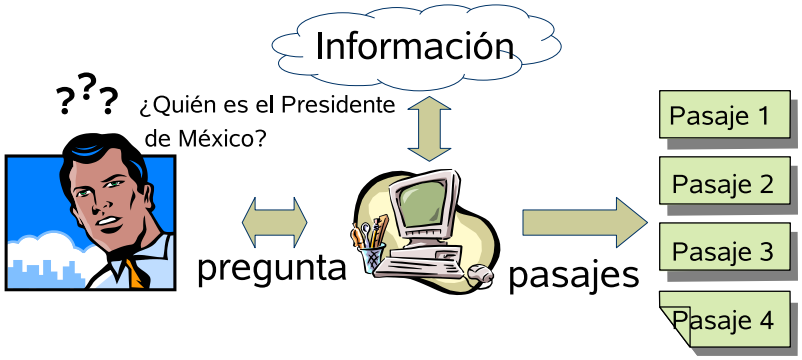


Figura 2.4: Sistemas de RP

Estos sistemas han demostrado mejorar considerablemente los resultados incluso en la RD ([LP01]). Su uso se ha extendido notablemente a otras áreas de la RI como es la BR. Esto es debido, fundamentalmente, a que estos sistemas permiten detectar extractos cortos de texto que son muy relevantes a pesar de que el documento, valorado en su conjunto, puede no ser considerado relevante por un sistema orientado a RD ([VILM03]). Además, en tareas como BR o EI, los sistemas de RP reducen enormemente la cantidad de texto necesaria para su posterior análisis.

2.3. El tamaño del pasaje

El tamaño del pasaje y la forma de extraerlo del documento es crucial para obtener buenos resultados. Los límites de los pasajes no están

explícitamente indicados en la colección de documentos. Los modelos de extracción de pasajes a partir de un documento dado normalmente se clasifican en modelos de ventanas, modelos de discurso y modelos semánticos ([Cal94]).

2.3.1. Modelos de ventanas

Los modelos de ventanas ([Cal94, JvRA+06]) se basan en crear pasajes de longitud fija. La técnica más sencilla es dividir el documento original en trozos de texto con un número fijo de palabras. Esta técnica, aunque simple y rápida, tiene el inconveniente de que divide el documento de forma muy aleatoria y, por tanto, puede separar palabras relacionadas.

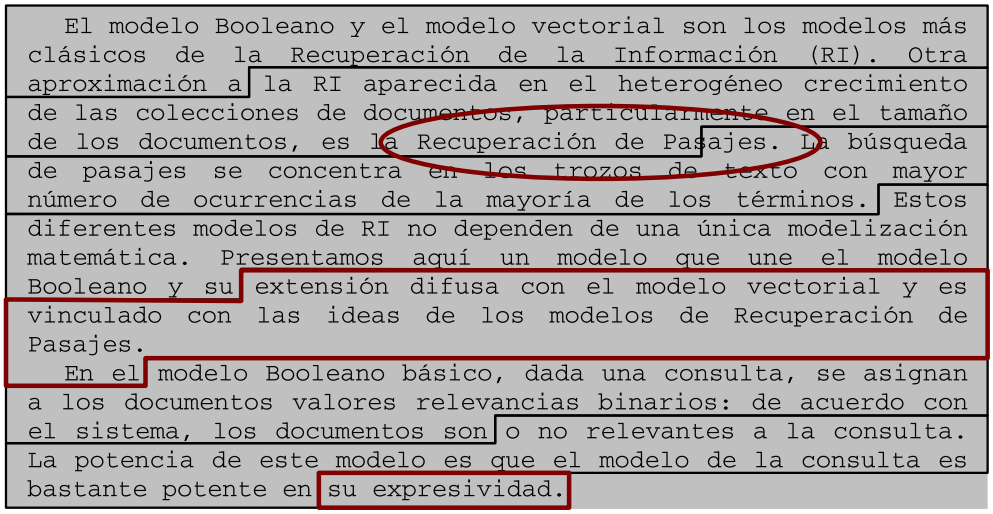


Figura 2.5: Modelo de ventanas de tamaño fijo de 20 palabras

En el ejemplo de la figura 2.5, si la consulta hubiera sido "*recuperación pasajes*", no se hubiera encontrado ningún pasaje con ambos términos aunque el documento sí que tuviera las dos palabras juntas.

Pese a que su principal ventaja consiste en obtener los pasajes del mismo tamaño, el último pasaje de cada documento posiblemente será menor que el resto puesto que no hay más palabras que añadir al pasaje.

Otros sistemas muy conocidos que usan métodos basados en ventana son: [CCPK99], [KZ97] y [HTC98].

2.3.2. Modelos de discurso

Los modelos de discurso ([Wil94, BMvN+06]) dividen los documentos a partir de las propiedades estructurales de éstos tales como secciones, párrafos o frases. La división por párrafos (figura 2.6) también es una técnica muy sencilla y rápida pero muchas veces el usuario utiliza los párrafos por razones de estética en vez de escoger una razón semántica ([Sal89, NIH+00]). Otro grave inconveniente que tiene esta división es que los párrafos tienen tamaños muy diversos y, por lo tanto, los pasajes que los contienen también.

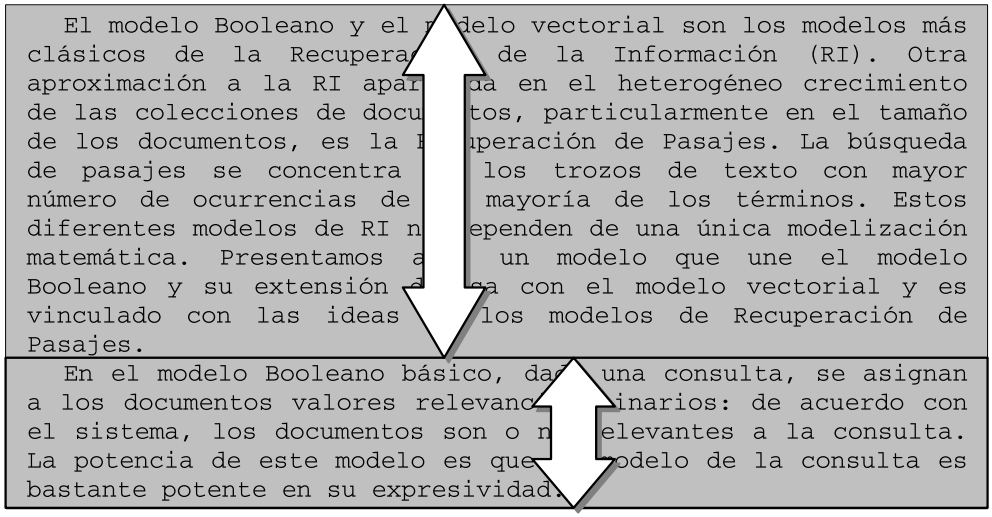


Figura 2.6: Modelo de discurso orientado a párrafos

Como se puede apreciar en la figura 2.6, se ha dividido el documento en dos pasajes. Cada pasaje contiene uno de los párrafos del documento, siendo el primero considerablemente mayor que el segundo en número de términos. Además, el segundo párrafo contiene información muy relacionada con el primero y la decisión de su división puede deberse a cuestiones estéticas. Tampoco existe un tamaño máximo o mínimo de

párrafo y, por esta razón, un párrafo podría contener todo un documento o, por otra parte, únicamente una frase.

2.3.3. Modelos semánticos

Quizás el mejor sistema para obtener los pasajes a partir de un documento sería aquel que, dado dicho documento, dividiese y agrupase el texto de tal forma que quedase pasajes cuyo texto contenga una semántica similar y relacionada con la pregunta [HP93, ZMWSD95, LSN06, MP06, SN06] (figura 2.7). Un parámetro configurable de este sistema sería el grado de similitud de la semántica intrapasaje. A pesar de ser un buen sistema, éste método tendría el inconveniente de que los distintos pasajes obtenidos serían de tamaños muy diversos y habría que compensar esto de alguna forma en la búsqueda de documentos relevantes. Sin embargo, estos modelos son muy difícil de estimar y muchas veces se incluyen tantos errores como aciertos.

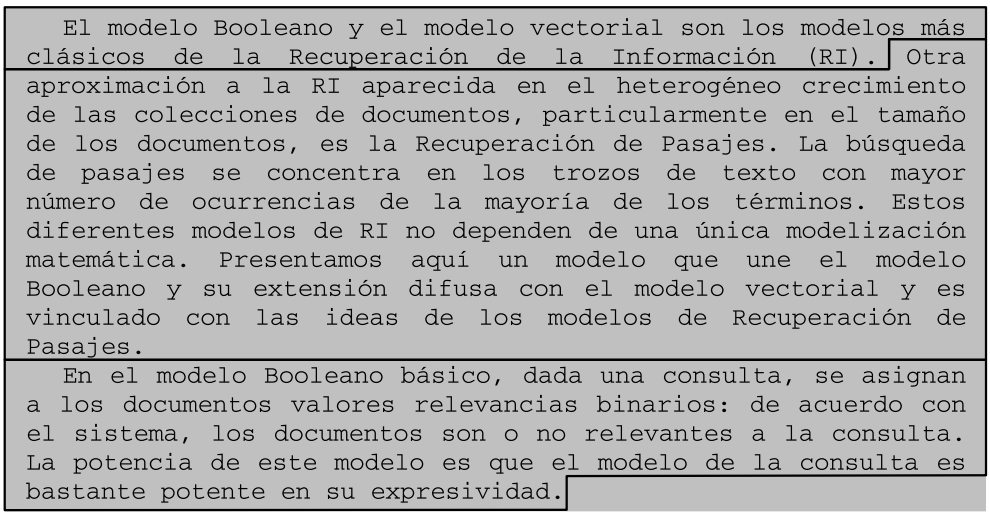


Figura 2.7: Modelo semántico

En el ejemplo de la figura 2.7, se han extraído tres pasajes. El primero habla sobre los modelos clásicos de RI como el modelo Booleano y el modelo EV. El segundo parece centrarse más en los sistemas de RI en

general. El tercer pasaje centra su atención sobre uno de los modelos de RI. En este ejemplo queda patente la gran disparidad de tamaños de cada uno de los pasajes. También es posible observar cómo esta división es muy subjetiva y depende de los criterios utilizados en la aplicación. Es más, estos criterios pueden depender, en gran medida, de la pregunta del usuario. Esto implicaría realizar la división por pasajes cuando el usuario lanza la pregunta, en vez de realizarla durante la indexación de la colección de documentos. La división semántica es un proceso lento y que requiere muchos recursos y, por lo tanto, puede no ser rentable a la hora de llevarla a cabo en tiempo real. La separación de pasajes del ejemplo ha sido realizada por una persona, un proceso automático, posiblemente, introduciría errores.

2.3.4. Comparativa entre modelos de división de documentos

En [Cal94] se hizo una comparación de estos métodos y concluyó que los métodos basados en ventanas dan un mejor resultados. Además, [KZ97] y [ZMWSD95] confirmaron los experimentos de Callan mostrando que los mejores modelos son los basados en ventana.

Posteriormente, [LVF02] unió dos métodos de extracción de pasajes: discurso y basado en ventanas. Utiliza una división de documento en el que la unidad de división es la frase en vez del término (figura 2.8). Aunque este método por lo general se incluye entre los métodos de discurso, también se podría considerar un modelo de ventana puesto que los pasajes son de una longitud de frases idéntica. Así, en su sistema llamado IR-n, tiene un parámetro n que determina de cuántas frases se compondrá el pasaje. La frase tiene un sentido semántico propio y, muchas veces, indivisible. Además, en un texto dado, las frases suelen ser de tamaño muy similar y no demasiado extenso. En su tesis [LP01], Llopis demostró que su sistema era, con diferencia, mejor utilizando las frases como unidad en vez de las palabras. Un inconveniente que tiene el sistema es que la división del texto en frases, a veces, no es tan evidente. Sus resultados mostraron que este tipo de método basado en ventana y en discurso mejora las precisión y el *recall* con respecto a los tradicionales sistemas de RP.

Un ejemplo de este modelo se puede observar en la figura 2.8. En este

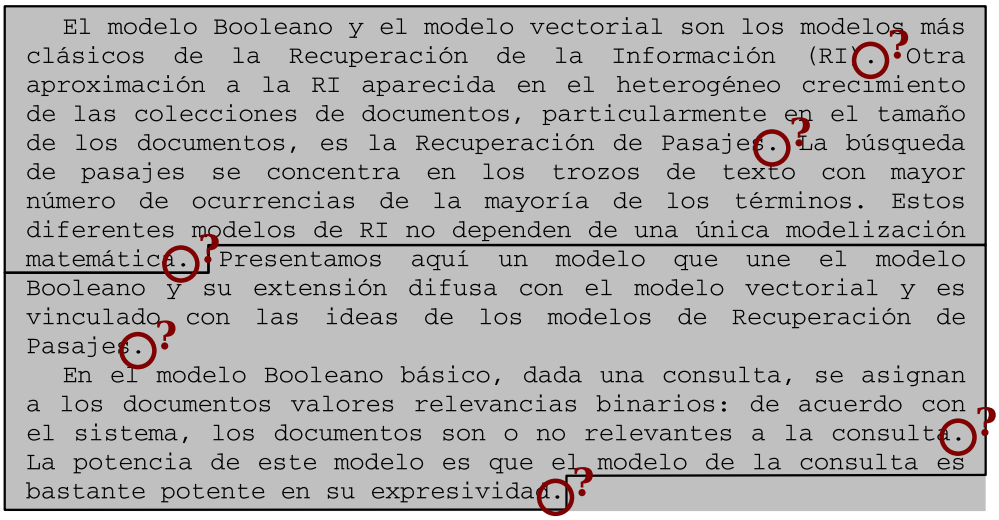


Figura 2.8: Modelo de discurso orientado a frases

caso hemos creado pasajes de tamaño de tres frases. Podemos observar que el tamaño de los pasajes es muy similar aunque hayan pequeñas diferencias de longitud en las frases. El documento ha sido obtenido al azar y ha dado la casualidad de que los pasajes son casi idénticos. Pero a veces se pueden crear pasajes con tamaños muy dispares. Además, al igual que ocurría con el modelo de ventanas, el último pasaje puede que no tenga el tamaño debido al no disponerse de tres frases al final del documento.

La separación por frases del ejemplo ha sido sencilla al terminar todas las frases con un punto y no haber puntos en otra parte. En documentos reales se puede encontrar siglas que terminan en puntos, o frases que no tienen un punto al final (como en el caso de los titulares de noticias de algunos periódicos). Además, este modelo se ha aplicado a un idioma en que casi siempre se terminan las frase con el mismo símbolo. Esto no ocurre en otros idiomas. En idiomas como el árabe, chino, hindi, etc. los símbolos de puntuación son distintos. También existen símbolos, como los de interrogación o admiración, que suelen terminar las frases y que se deben tener en cuenta para detectar sus límites.

Una ventaja de este modelo es que se pueden crear los pasajes de

forma dinámica durante el proceso de búsqueda ([GBBA+06]).

2.3.5. Solapamiento

Independientemente del modelo de extracción de pasajes utilizado, podría ocurrir que los términos de una consulta aparecieran en diferentes pasajes como se aprecia en la figura 2.9. Si en el documento del ejemplo buscásemos la consulta "*IR boolean model fuzzy*", nos encontraríamos, según el ejemplo, que el término "*IR*" aparece en el primer pasaje y el resto de los términos en el segundo pasaje.

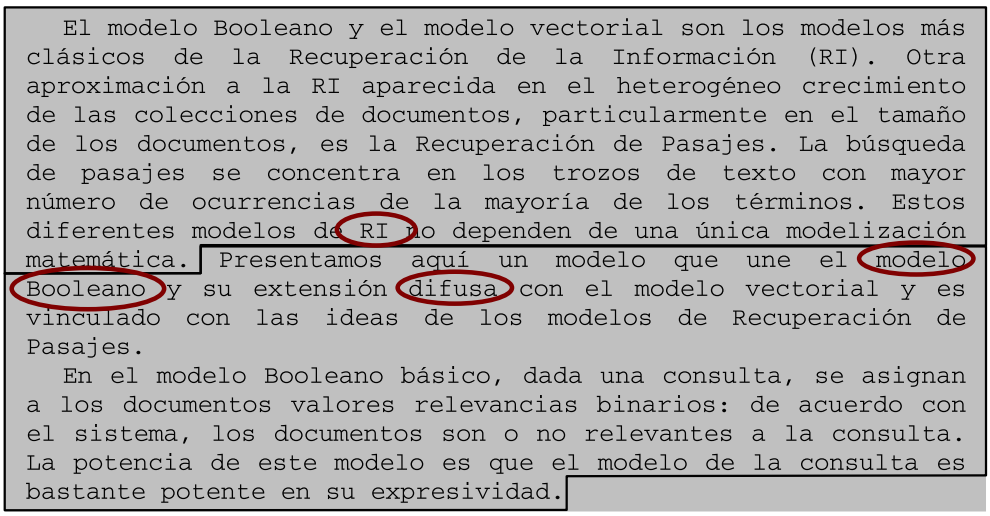


Figura 2.9: Pasajes sin solapamientos

Aún siendo un documento muy relevante para la búsqueda, no podríamos encontrar un pasaje en el que aparecieran todos los términos. Así, y para evitar este problema, se utiliza el solapamiento de pasajes ([LVF02, BYRN99]). El solapamiento hace que el pasaje siguiente empiece, no al final del anterior, sino solapándolo. Esta técnica se utiliza para minimizar el error cometido al dividir el documento y separar texto semánticamente relacionado. En los sistemas que se aplica el solapamiento contienen, típicamente, un parámetro que es el grado de solapamiento. El grado de solapamiento nos indica cuánto texto del pasaje anterior se solapará con el siguiente.

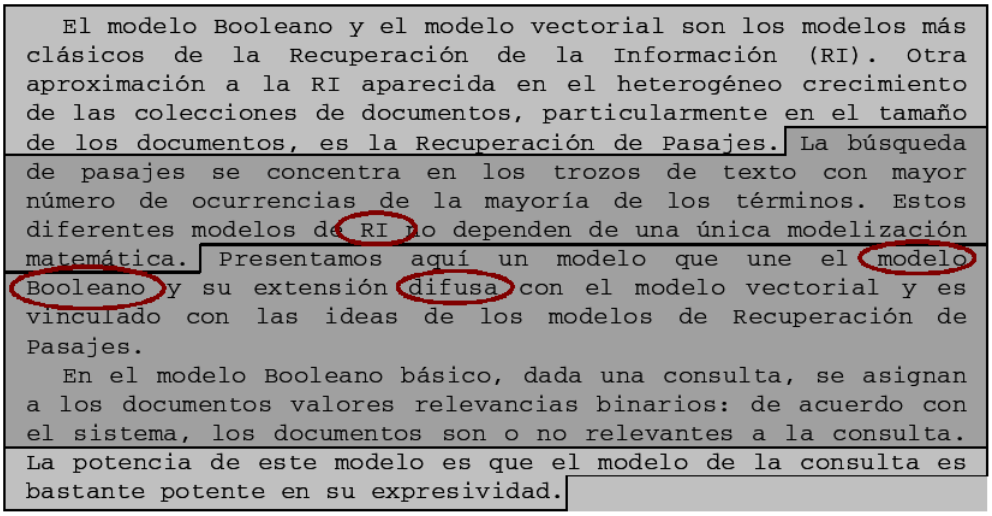


Figura 2.10: Pasajes con solapamientos

En el ejemplo de la figura 2.10 se puede apreciar que el pasaje de en medio, solapado con los otros dos, contiene todos los términos de la consulta original. De esta forma solventamos el problema que encontrábamos en la figura 2.9.

2.4. Sistemas de Búsqueda de Respuestas

Las técnicas actuales de RI han demostrado su éxito en grandes colecciones de documentos al determinar eficazmente el conjunto de documentos relevantes dada una petición de un usuario. Sin embargo, en ciertos escenarios el usuario desearía obtener una breve respuesta a una pregunta concreta. Por ejemplo, "*¿Quién descubrió América?*" o "*¿Cuántos astronautas han pisado la Luna?*". Actualmente, existen muy diversos esfuerzos enfocados a resolver esta problemática.

Los sistemas de BR intentan encontrar respuestas concretas a preguntas precisas formuladas por un usuario ([BCC+02]). Dada la complejidad de esta tarea, los sistemas actuales de BR sólo responden, hasta ahora, a peticiones factuales de un usuario casual. Es decir, se enfocan en responder preguntas simples sobre hechos concretos a partir de una colección

de documentos donde la respuesta se encuentra en forma explícita en un documento. Por supuesto, las capacidades de estos sistemas en el futuro permitirán resolver preguntas más complejas, incluyendo, por ejemplo, la de usuarios especializados con propósitos de análisis; además, de permitir responder a preguntas a partir de la fusión de la información de diferentes fuentes.

La investigación en sistemas de BR en dominios no restringidos inició su andadura tomando como punto de partida los logros obtenidos en sistemas de RI. En consecuencia, los primeros sistemas de BR disponibles utilizaron únicamente técnicas de RI en todos sus procesos. Como ejemplo, los sistemas de RMIT ([FKK+99]) y de las universidades de Waterloo ([CCPK99]) y Massachusetts (INQUERY) ([ACC+00]).

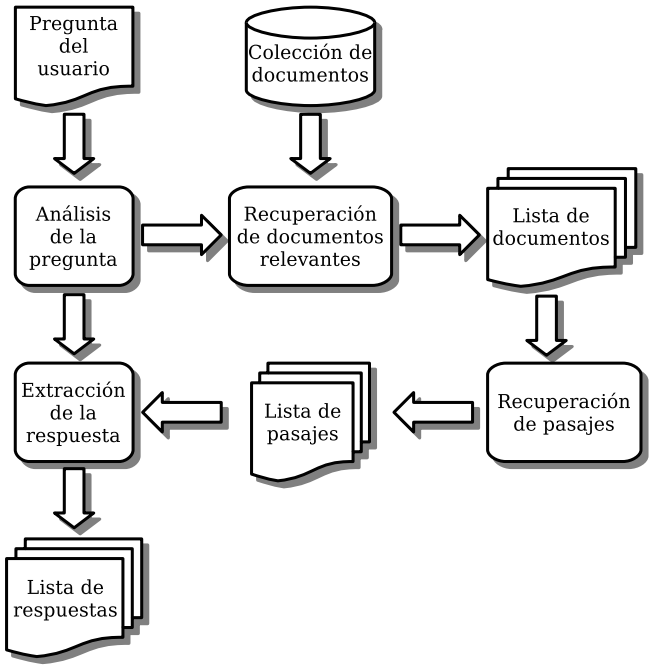


Figura 2.11: Típica estructura de un sistema de BR con modulo de RD

Los sistemas de BR típicamente pueden estructurarse en cuatro componentes: (i) el *análisis de la pregunta*, (ii) la *RD relevantes*, (iii) la *RP relacionados*; y (iv) la *extracción de la respuesta* (figura 2.12). El análisis

de la pregunta determina el tipo de respuesta esperado. Por ejemplo, el tipo de respuesta esperado para la pregunta "*¿Quién es el Ministro de Economía Alemán?*" es *persona*. Posteriormente, el segundo componente recupera los documentos relevantes a la pregunta. A partir de estos documentos, el componente de RP selecciona pequeños fragmentos donde es posible encontrar la respuesta. Básicamente, esta selección se apoya en la hipótesis de que los mismos términos usados para formular la pregunta son usados para expresar la respuesta. Cabe mencionar que existen sistemas de BR que incluyen uno o ambos componentes para la RI. Por supuesto, mientras más fina sea la granularidad se obtienen mejores resultados, de ahí que la mayoría de los sistemas usen módulos de RP. Por último, el componente de extracción de la respuesta busca entre los pasajes una posible respuesta a la pregunta en cuestión.

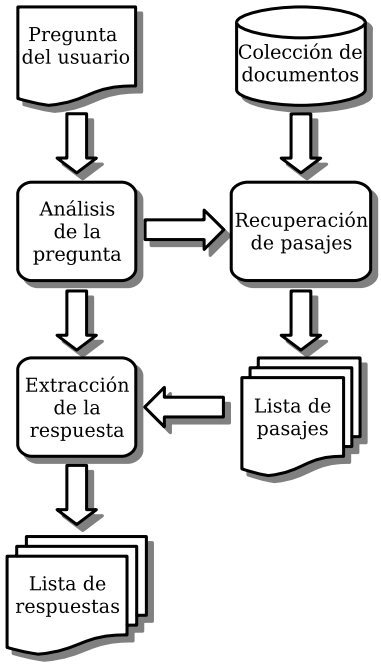


Figura 2.12: Típica estructura de un sistema de BR con módulo de RP

2.5. Análisis de la pregunta

En el análisis de la pregunta se obtiene información de la pregunta hecha por el usuario. Típicamente esta información consiste en las palabras claves y el tipo de la respuesta esperado ([BDB02, VILM03, CCL01, Buc01, DCEyGVP04]) pero otros sistemas más complejos también intentan obtener más información como el sentido sintáctico o semántico ([Lit99, CLMS+00, GH00, HEM02, JL02, HGH+00, MNPT02]) o realizar simples operaciones booleanas ([RFQ+02]).

El tipo de la respuesta esperado determina el grupo de reglas que se utilizarán posteriormente en la extracción de la respuesta. Este módulo tiene mucha importancia pues una mala clasificación de la pregunta puede llevar a aplicar reglas inadecuadas para extraer la respuesta. [MPHS03] mostró que el 36.4 % de los errores en los sistemas de BR se producen al extraer el tipo de la respuesta esperado.

Muchos sistemas de BR abarcan este problema mediante simples reglas ([VILM03, BLB+01, RFQ+02]). Si la pregunta empieza con un "*Cuándo*" entonces es una *fecha*, si empieza por "*Quién*" es una *persona*, etc. Este sistema funciona muy bien con preguntas del tipo "*Quién*", "*Cuándo*", "*Cuánto*" y "*Dónde*" pues son fáciles de clasificar, pero no ocurre lo mismo con las preguntas del tipo "*Qué*", ya que estas preguntas pueden abarcar desde una *persona* ("*¿Qué actor...?*") hasta *fechas* ("*¿Qué día...?*") pasando por "*cantidades*", "*organizaciones*", etc. Algunos incluyen etiquetas de partes de la oración para ayudar a la clasificación ([CLMS+00, Buc01]) y otros intentan abordar el problema mediante aprendizaje automático ([HGH+00]). También están los que utilizan análisis sintácticos y semánticos ([GH00, JL02]). Sin embargo, han habido aproximaciones que no utilizan ningún tipo de clasificación de la pregunta. [DCEyGVP04] extraen la respuesta sin tener en cuenta nada más que los términos más repetidos y ciertas heurísticas Ad hoc.

2.6. Sistemas de Recuperación de pasajes

Como hemos comentado anteriormente, el sistema de RP obtiene los trozos de texto relevantes, es decir, los pasajes, donde se creen que están las respuestas. El componente de RP es de crucial importancia porque

actúa como un filtro preliminar para reducir la colección de documentos original en un conjunto de pasajes en los cuales se buscará la respuesta. Esta reducción permite centrar la búsqueda eliminando la mayor cantidad de fragmentos irrelevantes posible. Por lo tanto, el rendimiento del sistema de RP determina el límite para el todo el sistema de BR ([Mon03]). Si el sistema de RP no recupera pasajes relevantes ningún proceso posterior podrá determinar la respuesta a la pregunta dada. Sería análogo al caso de formularle una pregunta a una persona que desconoce la respuesta.

Es importante contrastar la RD tradicional y la RP orientada a la BR. El primero de ellos, centra sus esfuerzos en encontrar documentos sobre el mismo tema de la petición, mientras que el segundo desea recuperar fragmentos de texto que contengan la respuesta de la pregunta dada ([Mon03]).

La RP orientada a la BR es más compleja que la RI tradicional por el simple hecho de que se trata de fragmentos de texto ([GBBA+06]). Básicamente, mientras más extenso sea el documento, más elementos se tendrán para determinar su pertinencia. Por el contrario, para determinar un pasaje como relevante se tienen menos elementos. En consecuencia, el número de documentos relevantes a un tópico particular es mucho mayor que el número de pasajes que contienen la respuesta a una pregunta dada ([Mon03]).

Por otro lado, un sistema de RP bajo el ámbito de la BR debe alcanzar niveles de cobertura y redundancia suficientes. Es decir, debe proporcionar pasajes relevantes para la mayor cantidad de preguntas formuladas y, además, debe proporcionar el mayor número de pasajes relevantes para cada pregunta ([CTCTC04]).

Para cubrir estos requerimientos se han propuesto diferentes métodos para establecer la similitud entre el pasaje y la pregunta. Entre estos métodos se destacan dos enfoques principales ([TKL+03]): (i) cuando el cálculo de la similitud depende del *solapamiento* entre los términos de la pregunta y los términos del pasaje, es decir, mientras mayor sea el solapamiento mayor será la similitud; (ii) cuando el cálculo de la similitud se basa en la *densidad* de los términos de la pregunta en el pasaje. Es decir, el método considera qué tan cerca aparecen, unas de otras, las palabras de la pregunta en el pasaje. Al comparar los sistemas de RP utilizados en el TREC, Tellex concluyó que los mejores métodos de recuperación de pasajes son los basados en densidad. Destacándose los métodos propuestos por

IBM ([IFZR00]) y por SiteQ ([LSL+01]). Por su parte, en la más reciente versión del CLEF, los sistemas de RP también pueden agruparse en estos dos enfoques, siendo el enfoque basado en solapamiento de términos ([BMvN+06, FKG+06, RFF+06, PSGLMF+06, SMS06, TVSI06]) más comúnmente usado que los basados en densidad ([BEF06, AFM+06]).

Cabe mencionar que en esta misma versión del CLEF se presentaron otros sistemas de RP que no caen completamente bajo alguna de estas dos categorías. Entre ellos se pueden mencionar a: (i) [AR06] quien además de considerar la cercanía de las palabras también considera el orden de aparición de los términos de la pregunta; (ii) [SMG+06] quien extiende el enfoque de solapamiento de términos al apoyarse en reformulaciones de la pregunta; (iii) [TKM+06] quien evaluó la similitud a nivel sintáctico a través del uso de la distancia de edición entre árboles; (iv) [Har06] quien transforma a una representación semántica la pregunta y los pasajes para determinar su similitud; (v) [Cos06] quien aplica una tipo especial de patrones léxico-sintácticos para determinar los pasajes relevantes.

En la mayoría de los sistemas de BR se utilizan clásicos sistemas de recuperación de pasajes ([MNPT02, AKM05, VILM03, NS05, OSS+05, MDVFCS05, GIM+05]). El principal problema es que los sistemas de RP que utilizan son adaptaciones de los sistemas de RD que se utilizaban en tareas clásicas de RI en vez de estar orientados a la específica problemática de la BR. Estos sistemas únicamente utilizan los términos claves de la pregunta para encontrar los pasajes relevantes. Por ejemplo, si la pregunta es "*¿Quién es el Presidente de México?*", estos sistemas devolverán pasajes que contengan los términos "*Presidente*" y "*México*". En [HGH+00, RG04] mostraron que los motores de RP estándares a menudo fallan al encontrar la respuesta en los documentos (o pasajes) cuando la pregunta está presentada en lenguaje natural. También en [GGH+03] se presentó un estudio del rendimiento de un sistema de BR usando los 20 primeros pasajes de cada pregunta y obtuvo la respuesta sólo en el 54 % del conjunto de preguntas.

Otras aproximaciones a la RP están basadas en el Procesamiento del Lenguaje Natural (PLN) ([Gre04, AAB+05, LC02]). Estas aproximaciones tienen la desventaja de ser muy difíciles de adaptar a otros lenguajes o a tareas multilingües. En la tarea de BR multilingüe es muy interesante usar metodologías de RP tan independientes del lenguaje como sea posible.

La estrategia de [DCEyGVP04, BLB+01, Buc01] se basa en buscar evidencias de la respuesta en la Web. Lanzan la pregunta del usuario a un buscador (usualmente Google[1]) con la esperanza de conseguir un pasaje que contenga la misma expresión que la pregunta o una similar. Suponen que debido a la alta redundancia de la Web, la respuesta debería estar escrita de todas las formas posibles, incluida la forma con que se ha formulado la pregunta. Para incrementar la posibilidad de encontrar los pasajes relevantes ellos hacen reformulaciones de la pregunta, es decir, mueven o borran términos de la pregunta para buscar otras estructuras. Por ejemplo, si movemos el verbo de la pregunta "*¿Quién es el Presidente de México?*" y borramos la palabra interrogativa "*Quién*" obtenemos la consulta "*el Presidente de México es*". Gracias a la reformulación y a la redundancia de la Web podríamos encontrar un pasaje con la estructura "*el Presidente de México es Vicente Fox*". Aunque [BLB+01] hace las reformulaciones llevando a cabo un análisis de las partes de la oración, [DCEyGVP04] hace las reformulaciones haciendo una serie de asunciones acerca de la posición del verbo y de los límites preposicionales de la frase en la pregunta. El problema de estos sistemas es que todas las posibles reformulaciones de la pregunta no pueden ser tenidas en cuenta dado el elevado coste de las búsquedas.

2.7. Extracción de la respuesta

Utilizando la información obtenida de la pregunta, se extrae la respuesta de los documentos o pasajes recuperados por el motor de búsqueda. Normalmente se buscan los términos candidatos de ser parte de la respuesta filtrando aquellos que no encajen con el tipo de la respuesta esperado, ya sea efectuando un análisis de partes de la oración ([Buc01]), sintáctico o semántico ([BD01]) o bien utilizando expresiones regulares ([VILM03, CLMS+00, MDVFCS05]). El aprendizaje automático también es utilizado para determinar el tipo de pregunta (o respuesta esperado) ([RFQ+02]).

Una vez obtenidos los candidatos, se selecciona la respuesta exacta buscando en bases de datos de conocimientos y ontologías ([RFQ+02, CLMS+00, HGH+00, MNPT02]) para comprobar si alguna respuesta can-

[1] http://www.google.es

didatas es la respuesta a la pregunta; o mediante métodos estocásticos y heurísticos utilizando la redundancia de la repuesta correcta ([VILM03, DCEyGVP04, CCL01]); o técnicas de aprendizaje automático; o realizando un análisis semántico ([GH00, HEM02, HGH+00, JL02, Lit99]). A las técnicas anteriores se les puede aplicar un análisis morfo-sintáctico para comprobar si el tipo del candidato coincide con el tipo de la respuesta esperado ([RFQ+02, VILM03]).

La Web es cada vez más utilizada para extraer la respuesta. A este respecto hay, actualmente, dos líneas de pensamiento principales. Los que buscan la respuesta en Internet primero y después la buscan en la colección de documentos para ver si la colección soporta la respuesta encontrada en Internet ([BLB+01, BDB02, HEM02]) o, a partir de los candidatos, la respuesta se valida con evidencias que se encuentran después en la Web ([VILM03]). [Buc01, DCEyGVP04] realizan la BR utilizando la Web como única fuente de información sin utilizar ninguna colección de documentos.

2.8. Campañas de evaluación de sistemas de Recuperación de la Información y Busqueda de Respuestas

2.8.1. Text REtrieval Conference

Aunque tradicionalmente el TREC trataba temas de RI y BR, actualmente abarca tareas más generales de RI. El TREC se celebra anualmente y pretende mejorar la investigación de la RI; descubrir nuevas líneas de investigación y aportar la infraestructura necesaria para llevarlas a cabo; aumentar la comunicación entre la industria, el mundo académico, y el gobierno mediante un foro abierto para el intercambio de ideas de investigación; aumentar la velocidad de transferencia de tecnología desde los laboratorios de investigación a los productos comerciales, demostrando que el aumento del rendimiento en problemas del mundo real mejora considerablemente cuando se usan las nuevas metodologías de la RI; e incrementar la viabilidad y el uso de las técnicas de evaluación apropiadas para la industria y el mundo académico, incluyendo el desarrollo de

nuevas técnicas de evaluación que se puedan aplicar mejor a los sistemas actuales de RI.

El TREC está patrocinado, principalmente, por el *National Institute of Standars and Technology (NIST)* que es un organismo americano que pretende desarrollar y promover medidas, estándares y tecnología para aumentar la productividad, facilitar el comercio y mejorar la calidad de vida. Es el mayor coesponsor de las conferencias del TREC, es el encargado de evaluar los resultados así como de preparar los recursos necesarios para que las tareas de investigación en cada área se lleven a cabo y es el encargado de dar soporte a los participantes de los TRECs.

Tracks del TREC

El TREC se divide en *tracks* o líneas de investigación que actúan como incubadoras de las áreas de investigación. En cada track se define claramente cual es el problema, se crea la infraestructura necesaria para soportar las tareas de investigación en ese área (colección de documentos de test, metodología de evaluación, etc.) y se crea una lista de correo para facilitar el intercambio de ideas. Algunas de estas áreas poseen página Web propia que se ubica, generalmente, en el servidor del NIST. Además de las líneas de investigación preestablecidas, existe un procedimiento para proponer nuevas líneas.

Las líneas de investigación existentes son las siguientes:

Blog Track Un track presente desde 2006 que pretende explorar la información presente en la blog-esfera.

Enterprise Track El objetivo de este track es estudiar la búsqueda aplicada al mundo empresarial, mejorando las RI sobre datos de empresariales.

Cross-Language Track Un track que investiga la posibilidad de los sistemas de recuperación para encontrar documentos que puedan ser relevantes para una consulta realizada por el usuario sin tener en cuenta el lenguaje en el cual fueron escritos (tanto la consulta como los documentos relevantes).

Este track posee su propia página Web ubicada en la dirección:

http://www.glue.umd.edu/ dlrg/clir/trec2002

Pero se centra en la recuperación de textos árabes e ingleses. Para el resto de documentos existe un foro en la página:

http://clef.ici.pi.cnr.it:2002/

Filtering Track Un track en el cual las preguntas que el usuario puede hacer al sistema no suelen variar y lo que cambia es la información de la colección de documentos. Así, estos sistemas, deben ser capaces de añadir información al sistema y que la nueva información añadida sea recuperada mediante cuestiones relacionadas.

Genomics Track El objetivo de este track es estudiar la RI en el área de la biología que estudia el genoma humano. No contiene únicamente la información sobre la secuencia genómica sino que también la documentación sobre la materia, como informes de investigación, informes de laboratorio, etc.

Dispone de una página Web en la dirección:

http://medir.ohsu.edu/ genomics

Legal Track Este track está orientado a la RI sobre documentos legales. Al igual que el *Blog Track*, LA *Legal Track* también está presente en las campañas del TREC desde 2006. Pese a ser una línea de investigación nueva dispone de la siguiente página Web:

http://trec-legal.umiacs.umd.edu

Million Query Track Este track intenta demostrar la hipótesis que una colección de prueba construida a partir de tópicos no completamente juzgados es un mejor corpus que los tradicionales colecciones del TREC. Su página Web reside en:

http://ciir.cs.umass.edu/research/million/

HARD Track El objetivo de HARD es llegar a conseguir mejoras en la RI mediante la aportación de información adicional acerca de la persona que busca y/o del contexto de búsqueda, a través de técnicas como la recuperación del pasaje, y usando una interacción con la persona que realiza una consulta muy dirigida. No se debe confundir esta línea de

investigación con la técnica de búsqueda de pasajes que se utiliza en los sistemas de recuperación de la información.

La página Web de este track se encuentra en:
http://ciir.cs.umass.edu/research/hard

Interactive Track Un track que estudia la interacción del usuario con los sistemas de RI. Participando los grupos de desarrolladores en un protocolo experimental consensuado y llevando a cabo estudios con usuarios reales.

La página Web es:
http://www.ted.cmis.csiro.au/TRECInt/

Novelty Track Un track para investigar la capacidad de los sistemas para colocar nueva información y evitar la información redundante.

Question Answering Track Un track diseñado para obtener una respuesta concreta, en lugar de una lista de documentos con rangos de relevancia, a una pregunta en lenguaje natural.

Tiene su propia Web en la dirección:
http://trec.nist.gov/data/qa.html

SPAM Track El creciente problema del SPAM ha impulsado este track que pretende mejorar las técnicas de filtrado de este tipo de correo electrónico.

Robust Retrieval Track Esta línea de investigación también conocida como *Ad Hoc* pretende investigar el rendimiento de los sistemas que buscan en conjuntos estáticos de documentos usando nuevas cuestiones o tópicos. La labor de este track es la tradicional tarea de RI Ad Hoc.

Curiosamente, esta línea de investigación no contiene página Web asociada. Esto es debido a que este track ha llegado a niveles de precisión tales que es muy difícil superar. Es una línea de investigación que no concursa desde el TREC-8.

Video Track Un track diseñado para investigar la recuperación basada en los contenidos de los digitales y conocimiento multimedia.

Su página Web es:
http://www.itl.nist.gov/iaui/894.02/projects/trecvid

Web Track Un track cuya tarea es buscar características en un conjunto de documentos que son una imagen (ejemplo o muestra) del World Wide Web.

Dispone de una página Web en la dirección:
http://es.cmis.csiro.au/TRECWeb/

Terabyte Track El objetivo de este track es investigar cómo los actuales sistemas de RI pueden ser escalados para soportar extensas colecciones de documentos. Su dirección Web es:
http://www-nlpir.nist.gov/projects/terabyte/

2.8.2. Cross-Language Evaluation Forum

El Cross-Language Evaluation Forum (CLEF)[2] apoya las aplicaciones que tratan de extraer información a partir de bibliotecas digitales mediante el desarrollo de una infraestructura para, primero, probar, poner a punto y evaluar los sistemas de RI que operan con los idiomas europeos en contexto monolingüe y cruce de idiomas y, segundo, crear conjuntos de prueba para utilizarlos de información que puede ser empleada por los desarrolladores de sistemas en sus evaluaciones.

Intenta crear una comunidad de investigadores y desarrolladores que estudien los mismos problemas y facilite la colaboración futura entre grupos con intereses similares. CLEF también establece fuertes enlaces y fomenta los intercambios de ideas y de resultados. El objetivo final es asistir y estimular a los desarrolladores de sistemas de RI multilingües además de garantizar sus competencia en el mercado.

Tracks del CLEF

Al igual que el TREC, el CLEF también se divide en *tracks* o líneas de investigación.

[2] http://clef.iei.pi.cnr.it/

Recuperación de la Información Monolingüe, Bilingüe y Multilingüe en Nuevas Colecciones El objetivo de este track es evaluar el rendimiento de los sistemas de RI en colecciones multilingüe para nuevos documentos. La colección de documentos para el CLEF contiene documentos en Inglés, Portugués, Inglés, Finlandés, Francés, Alemán, Italiano, Portugués, Español, Sueco, Ruso, Japón y Chino.

Recuperación de la información Monolingüe y de Cross-Language en Datos Científicos Estructurados (GIRT) El fundamento para este track es el estudio de la recuperación en un contexto de dominio específico usando la base de datos de ciencia social GIRT-4 Alemán/Inglés. Los datos de GIRT-4 son ofrecido como un corpus seudo-paralelo de Alemán e Inglés. El vocabulario multilingüe contenido (Alemán-Inglés, Alemán-Ruso) están disponibles. Se pueden realizar tareas Monolingües y Multilingües. Las consultas están disponibles en Inglés, Francés, Alemán y Ruso. Se espera también incluir nuevos datos científicos sociales (Francés, Ruso, Inglés).

Recuperación de Información Interactiva Multilingüe (iCLEF) El track del CLEF interactivo pretende estudiar el problema de la Búsqueda de Respuestas Multilingüe desde una perspectiva de usuario. Dependiendo de la perspectiva, el desafío es doble: desde el punto de vista de BR como una tarea mecánica, la interacción con el usuario puede ayudar al sistema BR a recuperar mejor las respuestas; desde el punto de vista de BR como tarea de usuario, un asistente investigador puede ayudar al usuario en la localización de la respuesta más rápidamente y de forma más sencilla.

Búsqueda de Respuestas Multilingüe (QA@CLEF) Se evaluarán los sistemas de BR Monolingüe y Multilingües. Los idiomas involucrados son el Holandés, Francés, Alemán, Italiano, Portugués, Español, Inglés. Casi todas las preguntas tendrán una respuesta en la colección de documentos, pero el conjunto de prueba incluirá también la definición de consultas y preguntas que no tienen una respuesta conocida en el corpus objetivo.

Recuperación Multilingüe en Colecciones de Imágenes (ImageCLEF) Este track evalúa la recuperación de imágenes descritas por títulos de texto basadas en consultas en un lenguaje diferente; ambos, el texto y la imagen emparejan técnicas que son potencialmente explotable. El track ofrece tres tareas: (1) una RI Ad hoc bilingüe, (2) una tarea de búsqueda interactiva, y (3) una tarea de recuperación en imágenes médicas. La primera tarea está también prevista como una tarea de niveles de entrada para los recién llegados al CLEF y al CLIR. Están disponibles dos colecciones de prueba: la colección de fotografía histórica de la Universidad de St. Andrew; y otra colección de imágenes médicas del Hospital Universitario de Geneva.

Recuperación de Documentación Multilingüe Transcrito (CL-SDR) Este track ayuda a la evaluación de sistemas CLIR de transcripción automática con ruido de documentos hablados (a partir de la colección del TREC SDR), y trabaja con los siguientes problemas: bilingüe SDR a partir del Holandés, Francés, Alemán, Italiano y Español; recuperación sin/con límites de historia conocida; uso de transcripciones automáticas múltiples.

Capítulo 3
Modelos de n-gramas

Como se ha mencionado en el capítulo 2, los actuales sistemas de BR utilizan modelos de RP que no están específicamente adaptados a las tareas de BR porque sólo tienen en cuenta las palabras claves para obtener los pasajes relevantes. Si uno de estos sistemas clásicos devolviera alguno de los pasajes que aparecen en la figura 3.1, éste sistema no sería capaz de discernir cuál de los dos pasajes presentados tiene mayor probabilidad de contener la respuesta puesto que ambos contienen las mismas palabras claves, en este caso "*capital*" y "*Croacia*".

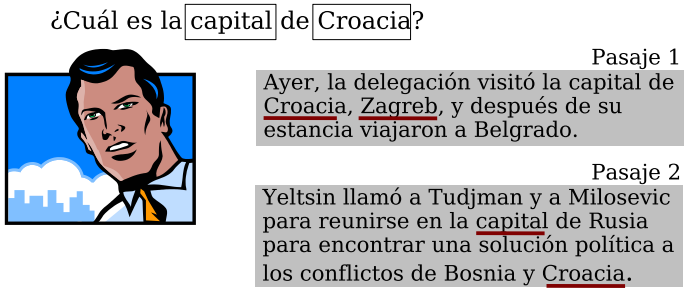


Figura 3.1: Ejemplo de dos posibles pasajes relevantes ante una pregunta del usuario.

JAVA Information Retrieval System (JIRS) es un sistema de RP orientado a BR cuya similitud, entre la pregunta y el pasaje, depende de la densidad y del orden de los términos de la pregunta en el pasaje, incluyendo tanto las palabras claves como las palabras comunes. Para realizar esta comparación, JIRS, en vez de buscar los términos de la consulta en la colección de documentos, busca las mejores estructuras, es decir, n-

gramas, que contiene términos o estructuras de la propia pregunta. Esto tiene como objetivo obtener aquellos pasajes con mayor probabilidad de contener la respuesta. Este sistema está basado en la hipótesis de que, en una colección de documentos suficientemente grande, siempre será posible encontrar una respuesta cuya sintaxis sea muy parecida a la expresión en que se formuló la pregunta. Es decir, JIRS aprovecha la *redundancia* de las grandes colecciones de texto y/o web para obtener no cualquier pasaje con las palabras claves sino aquellos pasajes con más alta probabilidad de contener la respuesta. Por ejemplo, si realizásemos la pregunta "*¿Cuál es la capital de Croacia?*" al sistema, éste intentaría devolvernos un pasaje con la expresión "*es la capital de Croacia*", con la esperanza que la respuesta aparezca como "***Zagreb*** *es la capital de Croacia*". Si no encontrase dicha expresión, devolverá, en primer lugar, el pasaje con la expresión más *similar* y, a continuación, el resto de pasajes a medida que la similitud se fuera reduciendo. Cada modelo de n-gramas tendrá una función de *similitud* distinta que medirá el grado de semejanza entre el pasaje y la pregunta. Estos modelos utilizan la misma estructura de la pregunta (incluyendo las palabras comunes) sin necesidad de tener un conocimiento previo de los idiomas con los que se trabaja, es decir, estos métodos son independientes del idioma.

JIRS es capaz de encontrar pasajes usando los n-gramas de la pregunta y calcular su similitud con el pasaje de una forma eficiente. Esto lo consigue gracias a una aproximación heurística. Se basa en utilizar un motor de RP clásico como primera etapa. Con este motor se extraen los n-gramas de los m primeros pasajes devueltos por el motor de búsqueda. Así, una vez obtenido estos n-gramas, se recalcula la similitud del pasaje comparando los n-gramas obtenidos con la pregunta, de tal manera que, cuanto más importante son los n-gramas encontrados, mayor similitud tendrá el pasaje. Se han desarrollado tres modelos para evaluar la importancia de cada n-grama, así como el peso final del pasaje y su similitud a partir del número de palabras que contiene, el peso de dichas palabras y la distribución de los n-gramas en el pasaje. A parte de estos modelos de n-gramas se han desarrollado dos motores de búsqueda, uno basado en el clásico modelo espacio vectorial y otro más dirigido a las necesidades de los modelos de n-gramas, el *Relevant Word Density* (RW-Density).

3.1. Arquitectura del sistema

JIRS puede adaptarse a diferentes arquitecturas dependiendo de la tarea que se vaya a realizar. Los cambios en su arquitectura son aplicados mediante cambios en su *Archivo de Configuración de Ejecuciones* o, de forma más dinámica, cambiando los argumentos del sistema[1]. En este apartado nos centraremos en las estructuras de JIRS que son utilizadas para la búsquedas por algún modelo de n-gramas.

Debido a que, para indexar la colección de documentos por n-gramas, es necesario conocer el tamaño máximo de las futuras preguntas a priori, no es factible realizar dicha indexación para cualquier tamaño de pregunta. Se puede aproximar una solución definiendo un tamaño máximo a los n-gramas que se pueden almacenar en el índice de ficheros invertidos. Esta aproximación tiene el inconveniente de que el fichero invertido necesario para almacenar los índices crecería exponencialmente a medida que estableciéramos un tamaño mayor de n-grama y a medida que aumentásemos la colección de documentos. Otra solución, que es la que hemos llevado a cabo, es indexar la colección a partir de sus términos y buscar los n-gramas en tiempo real durante la búsqueda del usuario. Esto nos evita tener que definir un tamaño máximo en la longitud de los n-gramas buscados. Además, nos permitiría utilizar sistemas de RI existentes como primera etapa. La arquitectura con la que JIRS realiza búsquedas utilizando modelos de n-gramas se representa en la figura 3.2.

La pregunta del usuario se pasa a un clásico sistema de RI basado en términos. Éste recupera sólo las frases donde alguno de los términos de la pregunta aparece. Se han adaptado varios sistemas de RI frecuentemente usados en tareas de BR en los foros de evaluación del TREC y CLEF. Los sistemas utilizados son el Lucene, MITRE y MULTITEXT. Estos sistemas vienen explicados en la sección 5.4.1 del capítulo 5. A parte de estos sistemas, se desarrolló un motor de búsqueda basado en el modelo EV mencionado en la sección 2.1.4 del capítulo 2. También se desarrolló un nuevo motor especialmente adaptado para suministrar a los modelos de n-gramas aquellas frases con un número mayor de términos relevantes de la pregunta. A este sistema se le denominó *Relevant Word Density*

[1] Cada tarea o ejecución distinta tiene sus propios parámetros que pueden ser cambiados como argumentos en la aplicación. Para más información al respecto véase apéndice B.

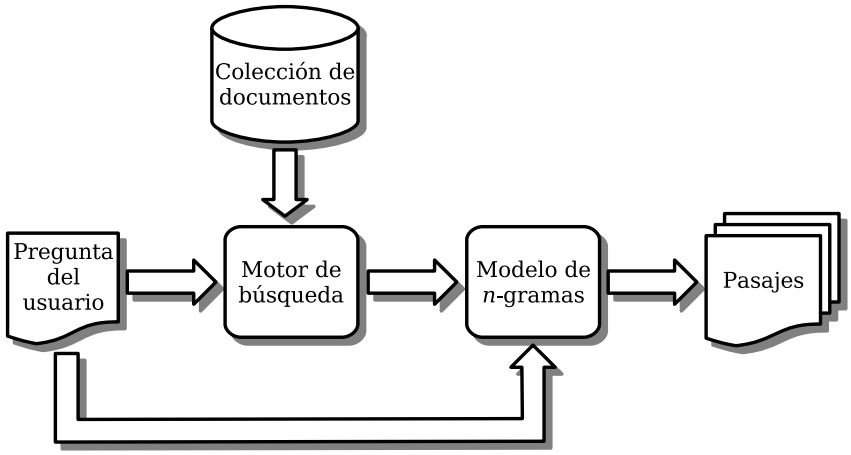


Figura 3.2: Arquitectura de JIRS para la búsqueda con modelos de n-gramas

(RW-Density).

El RW-Density no tiene en cuenta ningún otro factor en las frases devueltas mas que el peso de los términos de la pregunta que contienen. Así, para obtener la similitud entre un conjunto de términos relevantes de las frases f con respecto al conjunto q de palabras relevantes de la pregunta, el RW-Density la calcula como:

$$Sim(f,q) = \frac{\sum_{i=1}^{|q|} w_i \quad \text{if } t_i \in f}{\sum_{i=1}^{|q|} w_i} \tag{3.1}$$

donde $t_1, t_2, ..., t_{|q|}$ son los términos del conjunto q y $w_1, w_2, ..., w_{|q|}$ son sus respectivos pesos. El peso de los términos se calcula mediante la siguiente ecuación:

$$w_k = 1 - \frac{\ln(n_k)}{1 + \ln(N)} \tag{3.2}$$

Donde n_k es el número de frases de todo el sistema en los cuales el término t_k aparece y N es el número de frases del sistema. Esta función da mayor peso a aquellos términos que aparecen con menor frecuencia.

De tal manera que los términos que aparezcan una única vez tendrán un peso de 1 y los términos que aparezcan en todas las frases (como puede ser el caso de las palabras comunes), tendrán el mínimo valor que dependerá únicamente del número de frases del sistema $(1 - \frac{\ln(N)}{1+\ln(N)})$.

La razón de por qué les damos un peso pequeño a las palabras comunes es porque los modelos de n-gramas utilizan estos términos para formar los n-gramas. Como las palabras comunes no son indexadas se presume que estas palabras aparecen en todas las frases, así su n_k siempre será igual a N y, por tanto, su peso w_k siempre será el mínimo. Aunque el sistema RW-Density no utiliza estos términos al tener sólo en cuenta las palabras relevantes.

En pocas palabras, el RW-Density suma los pesos de los términos de la consulta que aparezcan en la frase y divide el resultado por el peso total de la pregunta para normalizar. Así, este método, fomenta aquellas frases que tienen más palabras relevantes y con mayor relevancia en una misma frase.

Con estos valores de similitud se ordena la lista de frases devueltos por el motor de búsqueda de mayor a menor similitud. Puesto que obtener los n-gramas de palabras de todo este conjunto de frases para ciertas preguntas puede ser muy costoso en tiempo y espacio, se escogen únicamente las m primeras frases y se descartan las otras. En el capítulo 5 veremos qué valor de m es el idóneo para cada modelo de n-gramas.

A continuación el modelo de n-gramas compara la consulta con respecto a cada una de las m primeras frases y reordena la lista devuelta por el motor de búsqueda según los nuevos valores de similitud asignados a las frases. Finalmente, se expande las frases con frases anteriores y posteriores para formar los pasajes que devolverá el sistema. Los modelos de búsqueda de JIRS permiten definir el número de frases anteriores y posteriores a añadir. En el capítulo 5 se realiza un estudio de cómo afecta este valor a los resultados.

3.2. Modelo de N-gramas Simple

Este modelo es el más simple puesto que no tiene en cuenta el peso de los términos que forman los n-gramas, sino el número de ellos. Su arquitectura es la que se muestra en la figura 3.3.

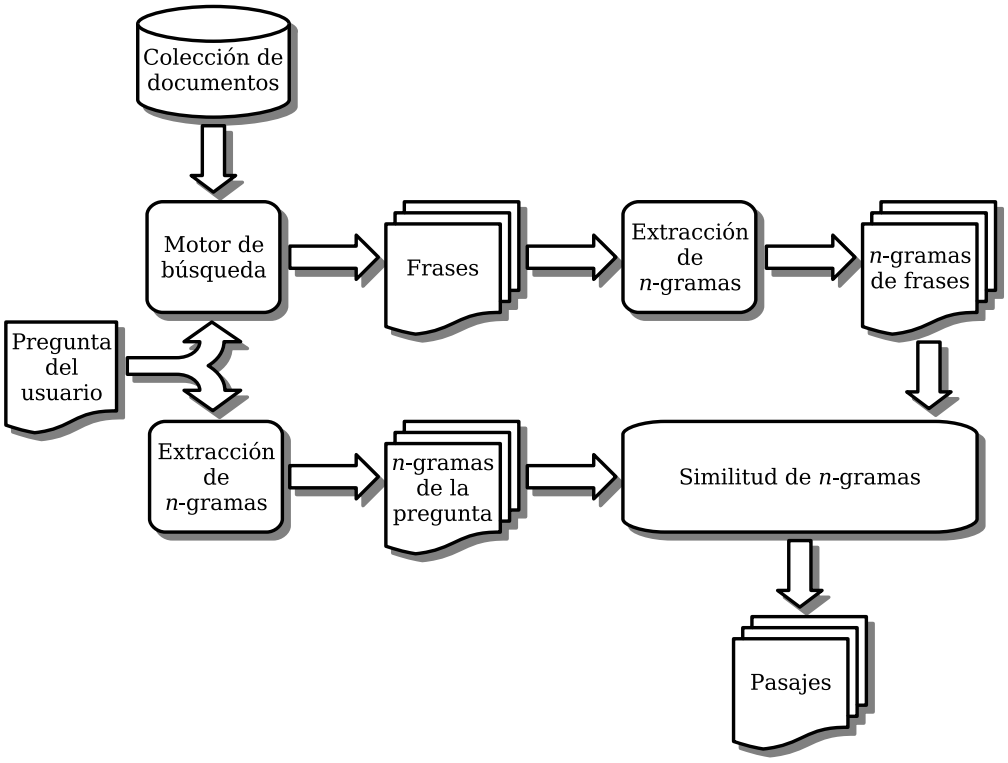


Figura 3.3: Arquitectura de JIRS para los modelos de n-gramas Simple y Term Weight

Este modelo comparte arquitectura con el modelo Term Weight que veremos en el siguiente apartado. En estos modelos, como se aprecia en la figura, la pregunta se entrega al motor de búsqueda y módulo de extracción de n-gramas. Como se ha mencionado en el apartado anterior, el motor de búsqueda devolverá frases de la colección de documentos donde algún término relevante de la pregunta aparezca y las reordenará según su criterio que dependerá del motor escogido. El módulo de extracción de n-gramas obtendrá todos los n-gramas de tamaño 1 hasta n, donde n es el número de términos de la pregunta (incluidas las palabras comunes). Aunque en este proceso se eliminarán las palabras interrogativas como "*Quién*", "*Cómo*", "*Qué*", etc. Un ejemplo de este proceso se puede observar en la figura 3.4.

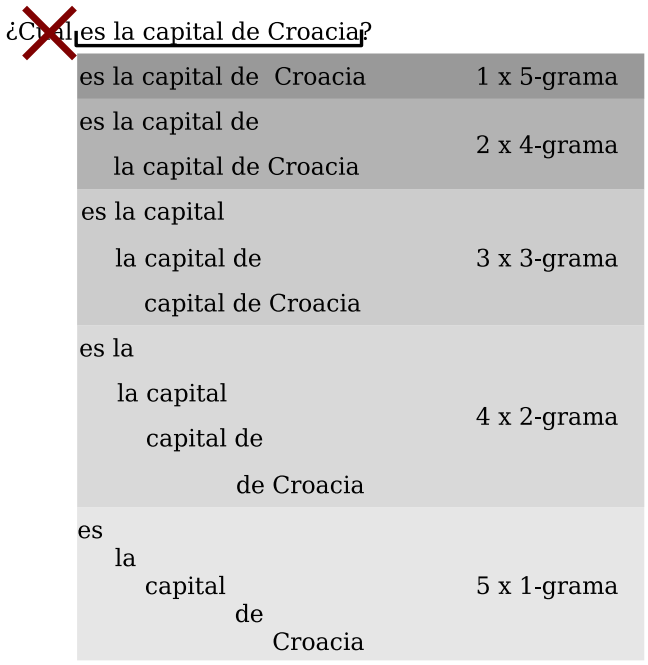


Figura 3.4: Ejemplo de la extracción de los n-gramas de la pregunta

Así, para la pregunta "*¿Cuál es la capital de Croacia?*", este módulo obtendría un 5-grama, dos 4-gramas, tres 3-gramas y así hasta cinco 1-gramas.

Una vez hemos obtenido los n-gramas de la pregunta, hacemos lo mismo con cada frase recuperada por el motor de búsqueda. Como el objetivo final de esto es comparar los n-gramas de la pregunta y de las frases, en el proceso de extracción de n-gramas de las frases sólo se tendrá en cuenta aquellos n-gramas que aparezcan en la pregunta.

De esta forma, y siguiendo con el ejemplo anterior, para las frases de la figura 3.5 obtendremos, para la primera frase, un 4-grama, dos 3-gramas, y así hasta cuatro 1-gramas. Sin embargo, para la segunda frase únicamente obtendríamos un 3-grama, dos 2-gramas y cuatro 1-gramas. Se puede observar claramente que, con esta información extra, es posible descubrir qué frase tiene mayor probabilidad de contener la respuesta. Cosa que no sería posible utilizando únicamente las palabras claves de

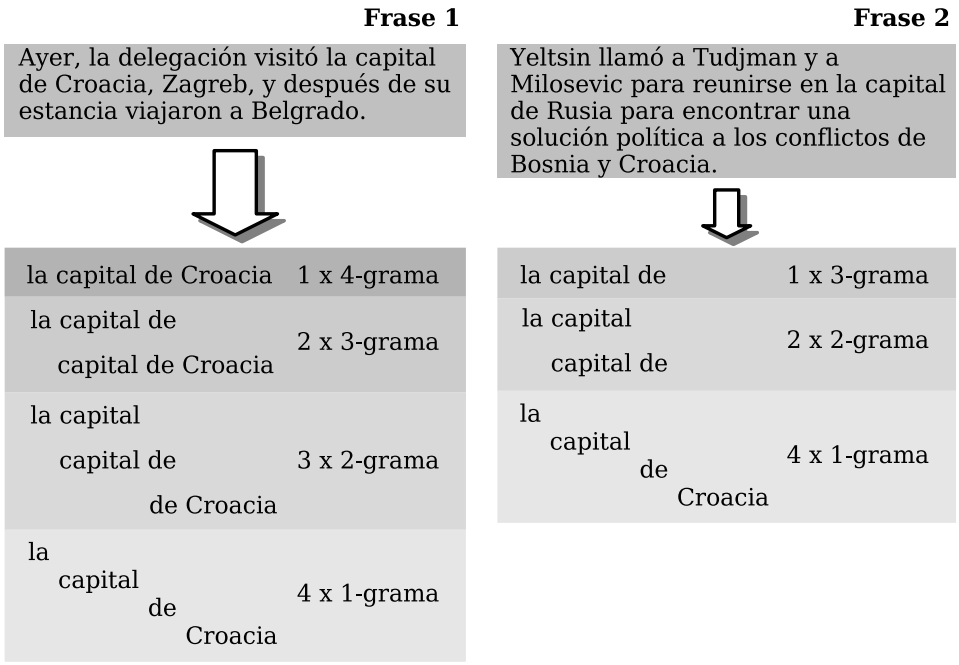


Figura 3.5: Ejemplo de extracción de los n-gramas en frases

la pregunta. Y todo esto sin tener en cuenta ningún aspecto del idioma más que una pequeña lista de palabras interrogativas.

Con los n-gramas de la pregunta y de las frases, se realiza una comparación para calcular un valor de similitud entre ellos. Este valor de similitud nos servirá para ordenar la lista de pasajes que finalmente será devuelta al usuario. En el modelo Simple, la similitud entre la pregunta y la frase recuperada se define como:

$$Sim(p,q) = \frac{\sum_{j=1}^{n} \sum_{\forall x \in Q_j} h(x, P_j)}{\sum_{j=1}^{n} \sum_{\forall x \in Q_j} h(x, Q_j)} \tag{3.3}$$

Donde $Sim(p,q)$ es una función que mide la similitud de los conjuntos de n-gramas de la pregunta q con respecto a los conjuntos de n-gramas de la frase p. Q_j es un conjunto de j-gramas que son generados a partir de la pregunta q, mientras que P_j es el conjunto de j-gramas de la frase

p con la que se quiere comparar. Es decir, Q_1 contiene los unigramas de la pregunta mientras que P_1 contiene los unigramas de la frase, Q_2 y P_2 los bigramas de la pregunta y de la frase respectivamente, y así hasta Q_n y P_n donde n es el número de términos en la pregunta.

La función $h(x, P_j)$ mide la relevancia del j-grama x con respecto al conjunto de j-gramas de la frase, mientras que la función $h(x, Q_j)$ es un factor de normalización porque el j-grama x siempre estará presente en el conjunto Q_j. En el modelo de n-gramas Simple, la función $h(x, P_j)$ se define como:

$$h(x, P_j) = \begin{cases} 1 & \text{if } x \in P_j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

En el modelo de n-gramas Simple cada n-grama de la pregunta que se encuentra en la frase tiene un peso de 1. Así, la similitud de una frase con respecto a la pregunta vendrá determinado por el número de n-gramas de la pregunta encontrados en la frase dividido por el número total de n-gramas.

En la figura 3.6 observamos los pesos de la pregunta y de las frases que se asignan utilizando el modelo de n-gramas Simple. Por lo tanto la similitud para la primera frase será igual a la suma de todos los n-gramas de la consulta que contenga dividido por el número total de n-gramas de la consulta (ecuación 3.5). Y si aplicamos la misma fórmula a la segunda frase obtendremos el resultado de la ecuación 3.6. Por tanto, para la primera frase, que es la que contiene la respuesta, obtendremos un valor de similitud mayor que con la segunda frase.

$$Sim(p_1, q) = \frac{10}{15} = 0{,}67 \tag{3.5}$$

$$Sim(p_2, q) = \frac{10}{15} = 0{,}47 \tag{3.6}$$

En este ejemplo podemos apreciar que ambas frases tienen diferentes valores de similitud aunque tengan las mismas palabras claves. Sin embargo, la diferencia entre los dos valores de similitud debería ser mayor. El siguiente modelo de n-gramas tendrá en cuenta la relevancia de los términos de los que está compuesto el n-grama.

¿Cuál es la capital de Croacia?	
es la capital de Croacia	1
es la capital de	1
la capital de Croacia	1
es la capital	1
la capital de	1
capital de Croacia	1
es la	1
la capital	1
capital de	1
de Crocia	1
es	1
la	1
capital	1
de	1
Croacia	+ 1
	15

Frase 1	
la capital de Croacia	1
la capital de	1
capital de Croacia	1
la capital	1
capital de	1
de Crocia	1
la	1
capital	1
de	1
Croacia	+ 1
	10

Frase 2	
la capital de	1
la capital	1
capital de	1
la	1
capital	1
de	1
Croacia	+ 1
	7

Figura 3.6: Ejemplo del modelo de n-gramas Simple

3.3. El modelo de n-gramas Term Weight

El modelo de n-gramas Simple tiene el problema que todos los n-gramas tienen el mismo peso. Por ejemplo, los n-gramas "*is the*" y "*the capital of Croatia*" del ejemplo de la figura 3.6. Esto puede causar que frases con n-gramas irrelevantes puedan ser más relevantes que otras con términos más importantes. Para resolver este problema hemos desarrollado el modelo de n-gramas de Term Weight. En este modelo el peso de un n-grama es dado por la suma de los pesos de los términos que contiene.

Así, la principal ecuación, que calcula la similitud entre la pregunta y las frases recuperados, es la misma que la del modelo de n-gramas Simple con la diferencia que la función $h(x, P_j)$ es calculada como:

$$h(x, P_j) = \begin{cases} \sum_{k=1}^{j} w_k & \text{if } x \in P_j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

Donde $w_1, w_2, ..., w_j$ son los pesos de los términos del j-grama $x = t_1 t_2 ... t_j$. Estos pesos deberían penalizar los términos que aparecen frecuentemente en la colección de documentos (por ejemplo, las palabras comunes) y promover las palabras relevantes pero tener un peso mínimo mayor de cero para las palabras comunes. Por esta razón hemos utilizado la función de pesado de términos de la ecuación 3.2.

Un ejemplo de este modelo se puede apreciar en la figura 3.7. En el ejemplo se han asignado unos pesos fictícios a los términos. Así, como se aprecia en el ejemplo, cada n-grama tendrá su propio peso, y la suma de los pesos de los n-gramas de cada frase marcará el peso total de la frase. Para calcular la similitud entre una frase y la pregunta, hay que dividir el peso de la frase por el peso de la pregunta, es decir, por el peso total de todos los n-gramas.

	0.1	0.1	0.2	0.1	0.5
	¿Cuál	**es**	**la**	**capital**	**de**	**Croacia?**
es la capital de Croacia						1
es la capital de						0.5
la capital de Croacia						0.9
es la capital						0.4
la capital de						0.4
capital de Croacia						0.8
es la						0.2
la capital						0.3
capital de						0.3
de Crocia						0.6
es						0.1
la						0.1
capital						0.2
de						0.1
Croacia						+ 0.5
						6.5

Frase 1	
la capital de Croacia	0.9
la capital de	0.5
capital de Croacia	0.8
la capital	0.3
capital de	0.3
de Crocia	0.6
la	0.1
capital	0.2
de	0.1
Croacia	+ 0.5
	4.2

Frase 2	
la capital de	0.4
la capital	0.3
capital de	0.3
la	0.1
capital	0.2
de	0.1
Croacia	+ 0.5
	1.9

Figura 3.7: Ejemplo del modelo de n-gramas Term Weight

En las ecuaciones 3.8 y 3.9 vemos los valores de similitud para las frases 1 y 2 respectivamente.

$$Sim(p_1, q) = \frac{4{,}2}{6{,}5} = 0{,}65 \tag{3.8}$$

$$Sim(p_2, q) = \frac{1{,}9}{6{,}5} = 0{,}29 \tag{3.9}$$

Con estos resultados podemos apreciar que, para el ejemplo de la figura 3.7, las diferencias de similitud entre ambas frases incrementa. Esto es porque la segunda frase no tiene n-gramas largos con los términos más pesados.

3.4. El modelo de Densidad de Distancias de N-gramas

En los modelos anteriores, las estructuras más largas continuaban teniendo mayor peso que las cortas, a pesar de que algunos n-gramas cortos tuvieran la mayoría de palabras claves y los largos no. Esto genera muchos problemas debido a la formación de n-gramas largos compuestos, en su mayor parte, por palabras comunes de la pregunta. Aunque el modelo Term Weight fue un intento de solucionar este problema veremos, en el capítulo 5, que su comportamiento es extremadamente parecido al modelo Simple. La causa es debida a la suma recursiva de los pesos de los n-gramas utilizada en el cálculo de la similitud tanto en el modelo Simple como el modelo Term Weight. Esta suma recursiva se produce puesto que un n-grama de tamaño n, tiene dos n-gramas de tamaño $n-1$, tres de tamaño $n-2$ y así hasta n de tamaño 1. Así, a la hora de calcular la similitud de la frase, se suman todos los pesos de estos n-gramas y, por lo tanto, el peso de un n-grama individual será igual al peso de dicha estructura más el peso de todas aquellas que contenga. Por ejemplo, la similitud de la frase con el n-grama "*capital de Croacia*" es la suma de los pesos de los siguientes n-gramas: "*capital de Croacia*", "*capital de*", "*de Croacia*", "*capital*", "*de*" y "*Croacia*". Esto también genera que la pérdida de un término poco importante en un n-grama de gran relevancia de tamaño n, reduzca la similitud de la frase enormemente al no contener dicho n-grama y parte de sus n-gramas de tamaño inferior que contenía.

Una posible solución sería eliminar la recursión en el cálulo de los pesos de las estructuras y sólo tener en cuenta los pesos de un n-grama,

sin sumar los n-gramas que contuviera. Esto no es factible ya que, en este caso, daría igual que los n-gramas estuvieran juntos, como en el caso de una frase que tuviera "*es la capital de Croacia*", o estuvieran separados, como, por ejemplo, una frase que tuviera todos los términos de la pregunta (incluyendo las palabras comunes) en diferentes partes del texto.

Otro problema de los modelos anteriores es que las reformulaciones de la pregunta encontrados en las frases, disminuyen en exceso la similitud de la frase. Es decir, si una frase tiene la expresión "*la capital de Croacia es*" su similitud es bastante menor que la expresión con el verbo al principio puesto que, el sistema, tomaría dicha expresión como dos n-gramas separados, uno con el texto "*la capital de Croacia*" y el otro con la palabra "*es*". Así, la similitud de una frase que podría ser importante se vería reducida enormemente.

El modelo de Densidad de Distancias de N-gramas (en adelante el modelo de Distancias), intenta resolver estos problemas. Para ello ha sido necesario modificar la arquitectura del sistema, que compartían los modelos Simple y Term Weight, por la de la figura 3.8.

En este modelo no se extraen los n-gramas de la pregunta para compararlos con los n-gramas de las frases, sino que obtenemos aquellas estructuras de las frases que estén compuestos por términos de la pregunta. Así, la pregunta del usuario, en este modelo, se pasará al motor de búsqueda clásico y, directamente, al modelo de Distancias. El motor de búsqueda, al igual que en los anteriores modelos, nos devolverá la lista de frases relevantes. Con cada frase de esta lista, obtendremos los n-gramas de mayor peso que estén compuestos por palabras de la pregunta (incluyendo las palabras comunes). En este caso, para formar los n-gramas, no nos importará el orden. En la figura 3.9 podemos ver un ejemplo de cómo funciona esta heurística.

En el ejemplo se observa cómo de la primera frase sólo se extrae el cuatrigrama "*la capital de Croacia*" en vez de extraer también el unigrama "*la*" que aparece al principio de la frase. Esto es debido a que el término de dicho unigrama ya es parte de un n-grama de mayor peso (en este caso del cuatrigrama) y, por lo tanto, ese término no se tiene en cuenta a la hora de formar otros n-gramas. Por otra parte, de la frase 2 extraemos dos estructuras principales: "*la capital de*" y "*Croacia*". Además de estas estructuras, se calculará la distancia de términos L entre cada n-grama

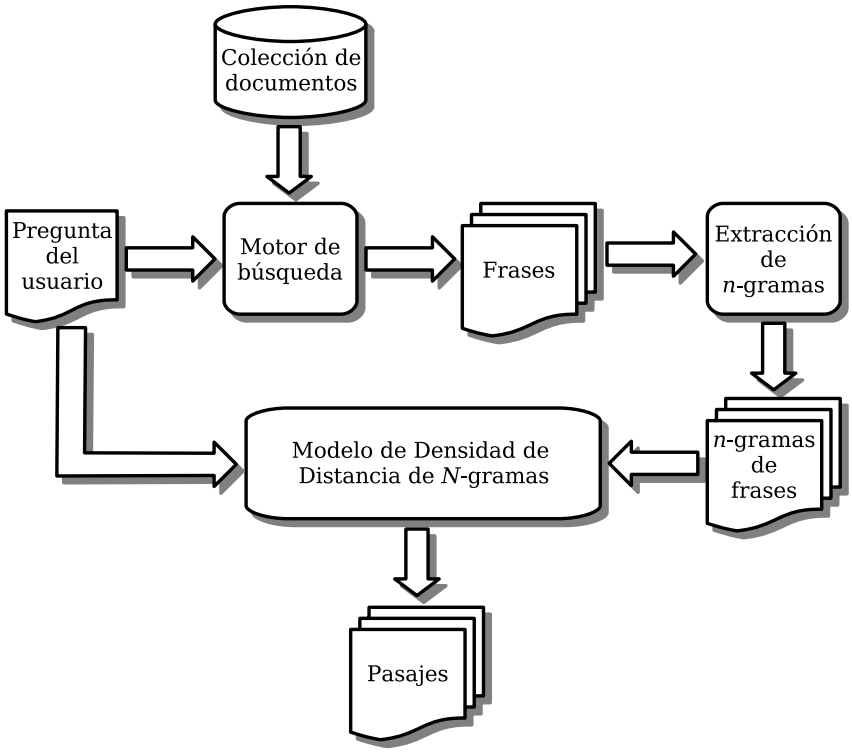


Figura 3.8: Arquitectura del modelo de Densidad de Distancias de N-gramas

y el n-grama de mayor peso de la frase. Por ejemplo, en la primera frase sólo hay un n-grama, por lo tanto, la distancia consigo mismo será de 0, mientras que en la segunda frase hat dos n-gramas, donde el n-grama "*la capital de*" está a 12 términos del más pesado. Esta distancia L se utilizará para calcular el factor de distancia. Este factor será igual a 1 cuando el n-grama esté *pegado* al más pesado e irá disminuyendo a medida que se amplie el número de términos que los separa.

En este modelo no buscamos las estructuras de la pregunta en las frases sino las estructuras de las frases compuestas por términos de la pregunta. De esta forma permitimos las reformulaciones de la pregunta. Por ejemplo, el n-grama "*la capital de Croacia es*" en este modelo tendría el mismo peso que "*es la capital de Croacia*" a pesar de que el primero no sea un n-grama de la pregunta y el segundo sí.

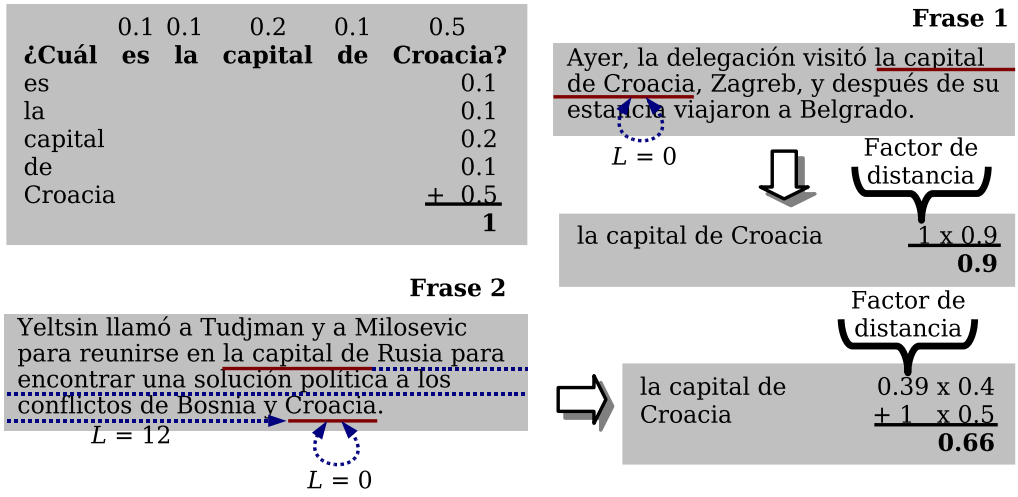


Figura 3.9: Ejemplo del modelo de Densidad de Distancias de N-gramas

Una vez extraído y pesado los n-gramas, éstos se envían al módulo de *Modelo de Densidad de Distancias de N-gramas*, que calcula la similitud de cada frase con respecto a la pregunta basándose en los n-gramas extraídos y en su disposición en la propio frase. El modelo de Distancias calcula la similitud de una frase mediante la siguiente fórmula:

$$Sim(p,q) = \frac{\sum\limits_{\forall x \in Q} h(x,P) \cdot \frac{1}{d(x,x_{max})}}{\sum\limits_{i=1}^{n} w_i} \tag{3.10}$$

Donde P es el conjunto de n-gramas de la frase p de mayor peso cuyos términos aparecen en la pregunta. En el conjunto P no puede haber más de un n-grama que comparta algún término. w_i es el peso del término i-ésimo de la pregunta y n es el número total de términos de la pregunta. La función $h(x,D)$, que mide el peso de cada n-grama, viene determinado por la ecuación 3.7.

$\frac{1}{d(x,x_{max})}$ es un factor de distancia que reduce el peso de los n-gramas que se encuentren lejos del n-grama de mayor peso. La función $d(x,x_{max})$ es la función que mide la distancia, es decir, el número de palabras, que hay entre un n-grama x y el n-grama de mayor peso x_{max}. Esta función

viene definida de la siguiente forma:

$$d(x, x_{max}) = 1 + k \cdot \ln(1 + L) \tag{3.11}$$

Donde k es un factor que determina la importancia de la distancia en el cálculo de la similitud. Con un valor pequeño la distancia afectará poco al peso de la frase y con un valor grande ésta será muy importante. L es el número de términos entre los n-gramas x_{max} y x.

Como vemos, el modelo de Distancias depende tanto de los n-gramas que aparecen en la pregunta, como de la densidad de éstos, es decir, una frase que tuviera muchos n-gramas relevantes y muy próximos tendría mayor similitud que otro cuyos n-gramas fueran menos y/o estuvieran más dispersos.

Como se ha podido observar en la figura 3.9, la similitud de la frase 1 es igual a la suma de los pesos de los términos del único n-grama extraído. Sin embargo, en la frase 2 se observa como el factor de distancia reduce el peso del trigrama "*la capital de*" puesto que se encuentra a 12 palabras del n-grama más pesado. Así, en esta frase, la similitud se verá reducida puesto que los dos bigramas encontrados se encuentran separados. Si no fuera por el factor de distancia, las dos frases tendrían la misma similitud porque ambas contienen los mismos términos de la pregunta.

En el modelo de Distancias, los pesos de los términos adquieren mucha importancia con respecto a los pesos de los n-gramas. Al contrario de lo que ocurría en los modelos Simple y Term Weight, un n-grama que esté compuesto únicamente por términos pocos relevantes o comunes, no modificará notablemente la similitud de la frase, sin embargo, si las palabras claves están concentradas en un único n-grama, la similitud será bastante alta, aunque éstos no incluyan algúna palabra común. Otro aspecto importante de este modelo es que la similitud de la frase no se ve afectada por las permutaciones de los términos de la pregunta que se produzcan en los n-gramas. Esta singularidad es muy importante en lenguajes cuyas frases que contienen la respuesta se formulan muchas veces como permutaciones de los términos de la pregunta. Por ejemplo, si preguntásemos "*¿Qué es la BBC?*, cabría esperar contestaciones del tipo "*la BBC es...*".

3.5. Reformulaciones de la pregunta

Los modelos de n-gramas se basan en la hipótesis de que la colección de documentos es suficientemente grande para contener alguna expresión con la respuesta que se parezca a la pregunta, independientemente de cómo sea la pregunta. Pero las colecciones de documentos no son tran grandes y, por lo tanto, no tienen la redundancia esperada. Además, muchas veces, estas colecciones están escritas con un estilo determinado, por ejemplo, el estilo periodístico que caracteriza las colecciones del CLEF. Esto impide que las respuestas se expresen de todas las formas posibles en un lenguaje dado. Por esto, para ciertos tipos de preguntas, es necesario realizar reformulaciones para que las consultas se ajusten más a cómo aparecen las respuestas en los pasajes. JIRS es capaz de realizar ciertas reformulaciones basadas en reglas a partir del tipo de respuesta esperado. La arquitectura del sistema con reformulación se puede apreciar en la figura 3.10.

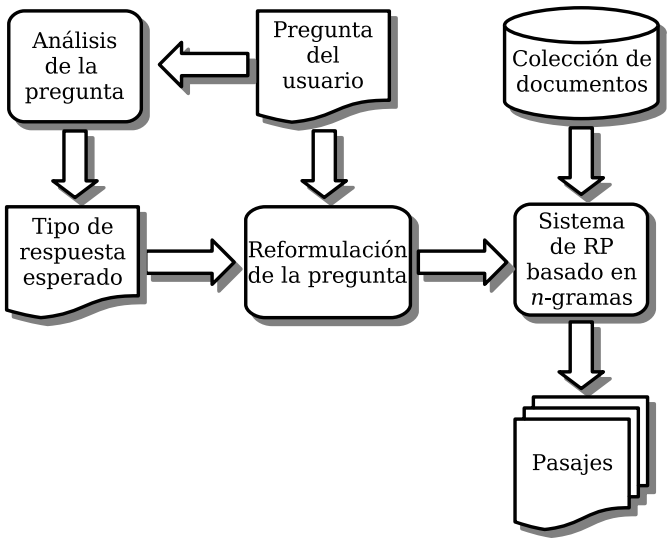


Figura 3.10: Arquitectura de JIRS con reformulación de la pregunta

En la nueva arquitectura, la pregunta del usuario, no va directamente al motor de búsqueda, sino que pasa primero por un módulo de análisis de la pregunta que obtiene el tipo de respuesta esperado. Con esta

información y la propia pregunta, el módulo *Question Reformulation* reformula la pregunta para dos tipos de preguntas. Montes-y-Gómez et al. [yGVPPC+05] obtuvieron muy buenos resultados en el concurso del CLEF en las preguntas de tipo definición[2]. Concretamente las preguntas sobre cargos de personas y el significado de las siglas de las organizaciones. Aprovechan el estilo periodístico de las colecciones de documentos utilizadas en el CLEF para buscar estructuras que aparecen muy frecuentemente cuando se define a una persona u organización. Se basaban en reglas y utilizaban dos patrones muy sencillos.

El primer patrón, que se utiliza para preguntas como "*¿Quién es Javier Solana?*", simplemente espera encontrar pasajes con la expresión "*, Javier Solana,*", es decir, que la persona que se busca aparezca entre comas. Esto permite encontrar pasajes que contengan la respuesta. Por ejemplo:

"*Para el* ***ministro de Asuntos Exteriores*** *, Javier Solana, es fundamental que...*"

El segundo patrón, utilizado para preguntas del tipo "*¿Qué es la ONU?*", buscan patrones en los que la siglas se encuentren entre paréntesis. Por ejemplo:

"*La* ***Organización de las Naciones Unidas*** *(ONU) o Naciones Unidas, es la mayor organización internacional del mundo.*"[3]

Con estos dos sencillos patrones y un buscador *Ad hoc* para encontrar las expresiones, Montes-y-Gómez alcanzarón precisiones de un 80 % de cobertura para este tipo de preguntas, casi el doble que la media.

El módulo de *Question Reformulation* de JIRS realiza las reformulaciones pertinentes para este tipo de preguntas. Así, los modelos de n-gramas no buscarán esctructuras con la sintaxis "*es Javier Solana*" o "*es la ONU*" sino que buscará pasajes con los siguientes n-gramas "*, Javier Solana,*" y "*(ONU)*", respectivamente.

[2] Para más detalles sobre los tipos de preguntas véase apartado 5.2.

[3] Extraído de Wikipedia (http://es.wikipedia.org/wiki/ONU).

3.6. Filtros de pasajes

Cuando se conoce el tipo de pregunta esperado es fácil eliminar pasajes, de la lista de pasajes devueltos por el sistema, que no cumplan ciertas condiciones. Por ejemplo, si lo que se espera como respuesta es una cantidad, podemos filtrar aquellos pasajes que no contengan cifras numéricas. Es más, si lo que precisamos es una cantidad monetaria, podemos refinar más el filtro eliminado aquellos pasajes que no contengan, a parte de una cifra, alguna unidad monetaria.

En la figura 3.11 se puede ver como hemos adaptado el filtro a la arquitectura de JIRS.

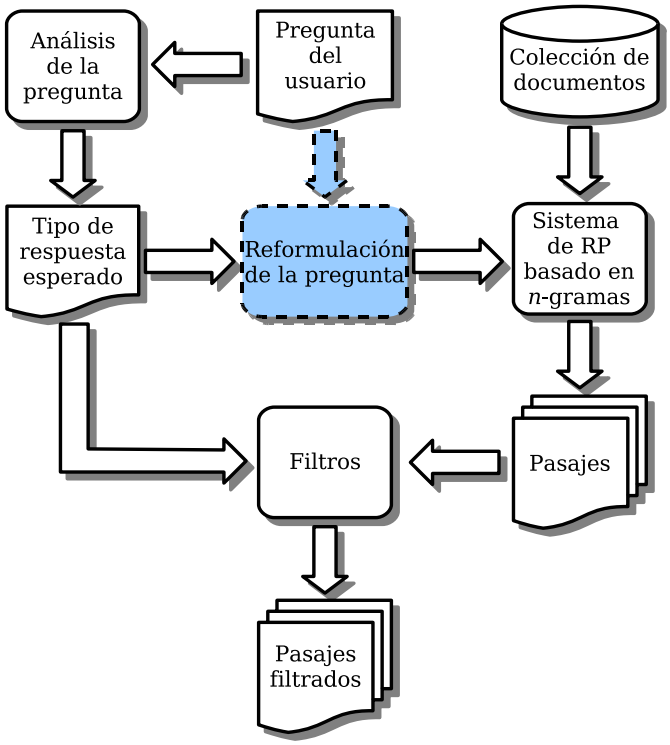


Figura 3.11: Arquitectura de JIRS con filtros y el múdlo de reformulación opcional

Los pasajes devueltos por el modelo de n-gramas se pasan al módulo *Filters* que elimina aquellos pasajes que no contengan información del

tipo que se espera como respuesta. No existen filtros para todos los tipos de respuesta puesto que nuestros filtros se basan en expresiones regulares y sólo hemos implementado para aquellos casos más sencillos. Los tipos de preguntas para los cuales se han aplicado filtros son:

DATE.YEAR Se filtran aquellos pasajes que no tengan cifras numéricas cuyo valor se encuentre entre 1200 y 1999 (puesto que la colección de documentos son colecciones de periódicos de 1994 y 1995).

DATE Se han filtrado aquellos pasajes que no contengan nombres de meses o número de años.

DATE.DAY Para pasajes con fechas completas que contemplen el día, mes y año.

QUANTITY.MONEY Sólo pasan el filtro aquellos que contengan nombres de monedas.

QUANTITY Cualquier cifra aunque no se exprese numéricamente.

QUANTITY.DIMENSION Se filtran los pasajes que no tengan unidades de medida (metro, kilográmo, etc.).

QUANTITY.AGE Sólo aquellos pasajes que contengan palabras como años, siglos, milenios, etc. pueden pasar el filtro.

Además de estos filtros que dependen en gran medida del tipo de respuesta esperado, se aplica un último filtro, para cualquier tipo de pregunta, que filtra por palabras *pivots*. Para este filtro, las palabras pivots son aquellas palabras que empiezan por mayúscula o por un dígito. Nos hemos dado cuenta que, para las preguntas del CLEF, este tipo de palabras deben aparecer en el mismo pasaje que la respuesta. Si no, o bien no se encuentra la respuesta, o bien la respuesta no está soportada, es decir, que en el pasaje no contesta explícitamente a la pregunta aunque la respuesta aparezca. Este filtro es necesario debido a que JIRS, a veces, devuelve pasajes que no contienen alguna de las palabras pivots pero sí otros términos de la pregunta. Por esto, pueden existir muchos pasajes en la lista de pasajes devueltos que no contengan estas palabras *pivots* y, por lo tanto, hayan muchos pasajes que no sean relevantes a la consulta. De esta forma eliminamos, de forma muy sencilla, muchos pasajes ruidosos.

Tanto el módulo de reformulación como el de filtro, no son independientes del idioma y hacen que JIRS tampoco lo sea. Además, tanto la reformulación como los filtros, sólo están implementados para ciertos tipos de preguntas y en ciertos idiomas. Jugando con los parámetros de JIRS se puede activar o desactivar tanto las reformulaciones como los filtros de forma separada. Es decir, que se puede activar el módulo de reformulación pero no el de filtros, se puede activar el de filtro pero no el de reformulación, y se pueden activar o desactivar los dos conjuntamente. Además, JIRS, cuando trabaja con estos módulos, es capaz de detectar el idioma con el que se está trabajando y no aplicar los módulos que no soporten el idioma correspondiente.

Capítulo 4
Descripción del sistema

4.1. Arquitectura del sistema

JAVA Information Retrieval System (JIRS) es un sistema de RI y RP de alta modularidad, escalabilidad y configuración. A parte de realizar búsquedas por los tradicionales métodos de búsquedas basados en términos, permite hacer búsquedas basadas en n-gramas. Esto lo hace especialmente apropiado para sistemas de BR multilingüe. Está escrito integramente en JAVA. La estructura general de la aplicación se muestra en la figura 4.1.

JIRS se compone de un núcleo llamado Java Process Manager (JPM), unos archivos de configuración y un conjunto de herramientas para procesar y buscar texto. Además, está preparado para incluir nuevos procesos de forma extremadamente fácil. JPM es un gestor de procesos que permite añadir o modificar la operatividad del sistema así como los parámetros de ejecución de una forma sencilla sin recompilar toda la aplicación, únicamente modificando los archivos de configuración. Dichos archivos tienen una estructura jerárquica basada en documentos XML que permite estructurar la información de una forma lógica. Los archivos de configuración no se componen únicamente de parámetros de la forma nombre-valor que determinan la configuración de las diferentes *acciones*, sino que determinan qué acciones y cuál será el orden de ejecución de dichas acciones. De esta forma se puede modificar totalmente el comportamiento del sistema cambiando únicamente el archivo de configuración.

Todo el código se ha comentado utilizando la herramienta de SUN *javadoc* que permite la documentación automática del código. Aporta diferentes herramientas de programación en las que se incluyen métodos

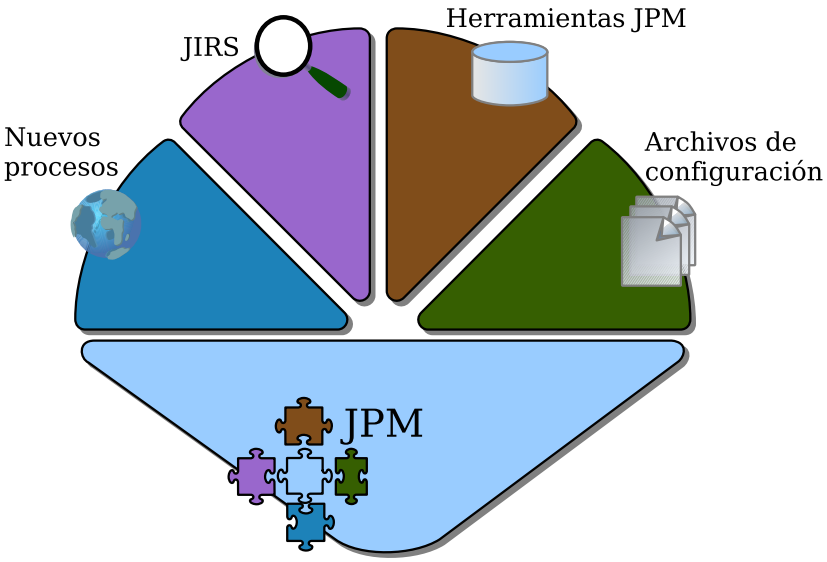


Figura 4.1: Arquitectura general de JIRS

para visualizar el progreso de los procesos que forman parte de una ejecución JIRS, imprimir mensajes *verbose* a distintos niveles, obtener la configuración para cada proceso o método, ejecutar otros subprocesos o métodos, etc.

Las principales características del sistema son:

- Una estructura jerarquizada de los parámetros, ejecuciones, procesos y métodos, tanto a nivel de configuración como de implementación.
- La configuración se realiza sobre archivos XML.
- Una comunicación rápida y eficaz entre procesos.
- Para añadir operatividad se añade las nuevas clases dentro de la jerarquía de paquetes del sistema y se añade las entradas concretas en el archivo de configuración, sin tener que recompilar toda la aplicación.
- Gran flexibilidad a la hora de configurar las distintas acciones del sistema.

- Los procesos de un mismo nivel se encadenan automáticamente, tanto secuencialmente como de forma paralela.
- Permite una estructura cliente/servidor para realizar procesos.
- Los procesos se pueden ejecutar de forma paralela y distribuida, JIRS se encarga de sincronizar los diferentes procesos concurrentes automáticamente independientemente de si se ejecutan localmente o remotamente.
- Se aportan una serie de herramientas de programación para generar los nuevos procesos y métodos.
- Al estar basado en el lenguaje JAVA el sistema adquiere todas las propiedades de éste, como son: multiplataforma, documentación automática del código, minimización de los errores de programación, entorno de ventanas swing o awt, etc.

JIRS incluye tanto bibliotecas de clases orientadas a sistemas de RI como bibliotecas genéricas que contienen clases útiles para otras tareas. La gran escalabilidad y configuración de JIRS la aporta el JPM.

4.2. Java Process Manager

Toda la flexibilidad de JIRS se debe a su núcleo. JPM es el núcleo que permite crear una aplicación a partir de un archivo XML. En este archivo de configuración se definirán los parámetros, procesos y métodos que se utilizarán en una o varias ejecuciones. Así, una ejecución o *run* contendrán parámetros (*parameters*), procesos (*processes*) y métodos (*methods*) que le indicará qué tarea realizará y con qué configuración. En la figura 4.2 se puede ver un ejemplo de una ejecución mediante JPM.

A partir del archivo de configuración, el JPM selecciona una de las ejecuciones del archivo y ejecuta los procesos que contenga. En el ejemplo de la figura 4.2, el proceso 1 se ejecuta y, como resultado, genera un objeto que es enviado al proceso 2. Éste, a su vez, realiza operaciones con este objeto y genera otro que, en vez de pasarlo al proceso 3, se lo envía a sus subprocesos, concretamente al primero. Este subproceso vuelve a

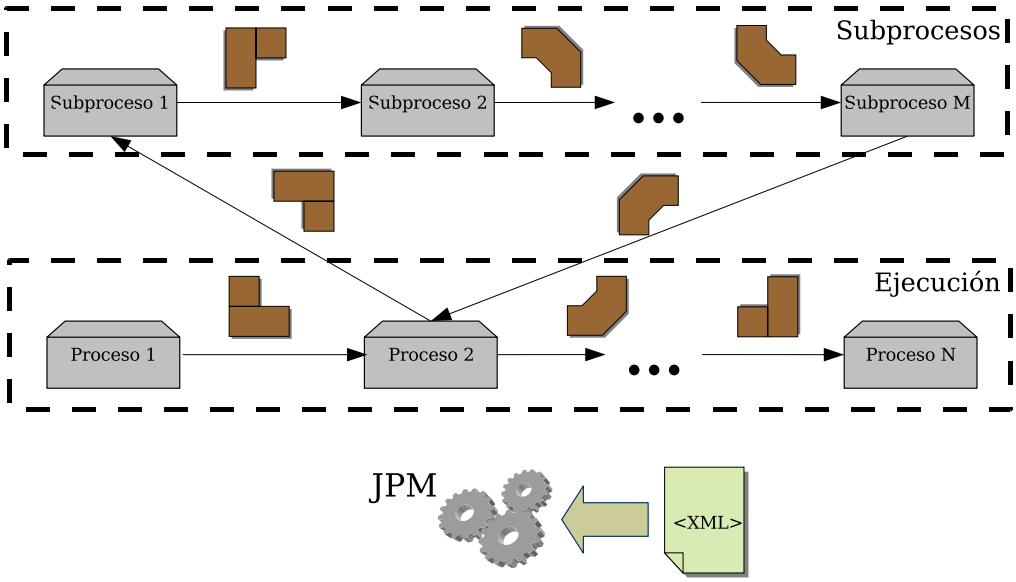


Figura 4.2: Ejemplo de una ejecución mediante el JPM

realizar operaciones con este objeto y pasa uno nuevo al siguiente subproceso. Esto se repite mientras queden procesos y subprocesos hasta que, finalmente, entrega un objeto de vuelta al proceso 2 que podrá procesarlo o entregarlo a los siguientes procesos. Esto se realiza hasta finalizar todos los proceso de la ejecución.

Éste es un ejemplo simplificado de las posibilidades del JPM pues, sólo se han visto implicados procesos y subprocesos. Pero esta estructura puede ser cambiada a partir del archivo de configuración para crear aplicaciones más complicadas que contengan procesos paralelos, arquitecturas cliente/servidor, distribuidas, etc. En el siguiente apartado se explica con más detalle el archivo de configuración de ejecuciones.

4.3. Archivo de Configuración de Ejecuciones

Como se ha mencionado anteriormente, la aplicación JPM necesita un archivo de configuración que le indique qué ejecuciones se podrán llevar

a cabo, qué procesos y subprocesos tendrán dichas ejecuciones y con qué parámetros y métodos se lanzarán.

Estos archivos de configuración están escritos en XML y cada uno de éstos dispone de un identificador y un título. En el nivel más superior de este documento XML se puede definir diversas ejecuciones, así como procesos, métodos y parámetros globales. En el siguiente código se muestra la arquitectura general de este archivo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE runs SYSTEM "runs.dtd">
<!-- JIRS default configuration -->
<runs name="DEFAULT" title="Default configuration">
  <!-- Global parameters -->
  <parameters>...</parameters>
  <!-- Global methods -->
  <methods>...</methods>
  <!-- Global processes -->
  <processes>...</processes>
  <!-- Example of one execution -->
  <run name="Execution 1" title="Example of first execution">...</run>
  ...
  <!-- Example of other execution -->
  <run name="Execution n" title="Example of last execution">...</run>
</runs>
```

Las dos primeras líneas indican que el archivo será un documento XML y se definirá qué etiquetas XML estarán permitidas. Todo archivo de configuración tendrá un elemento raíz llamado *runs* en el cual se definirán los parámetros, métodos, procesos y ejecuciones globales. Este elemento tiene dos atributos *name* que es el identificador de este archivo de configuración y `title` que es el título que tendrá el documento. El valor de estos atributos se deja a discreción del usuario, aunque es recomendable que el atributo *name* sea único entre los archivos de configuración de ejecuciones del sistema.

4.3.1. Acciones

Una acción en JPM es una clase abstracta de la cual se hereda las ejecuciones, procesos y métodos que pueden ser definidos en el archivo de configuración. Gracias a esta jerarquía existen muchas similitudes entre los tres tipos diferentes de acciones como la posibilidad de enviar mensajes de depuración, ejecutar subprocesos y métodos, interrogar al sistema

por un parámetro, etc.

4.3.2. Ejecuciones

En un archivo de configuración se puede definir una o más ejecuciones que serán las encargadas de definir las tareas globales que podrá hacer el sistema. En la figura 4.3 se puede observar el ejemplo de la ejecución *SearchClient*.

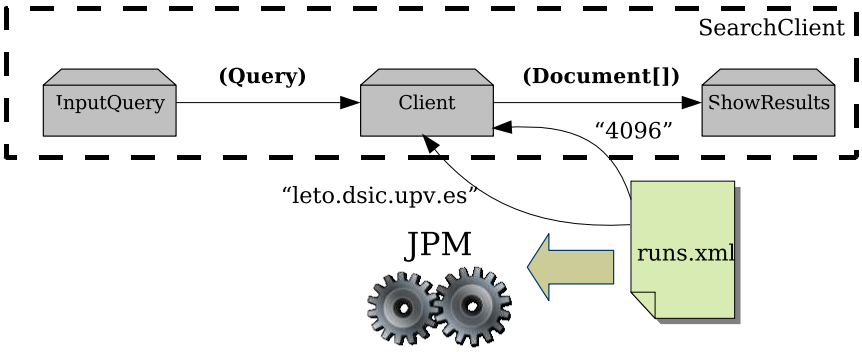


Figura 4.3: Ejemplo de una ejecución

El usuario únicamente debe indicar al JPM qué ejecución debe lanzar y el JPM ejecutará sencuencialmente los procesos que la ejecución *SearchCliente* contenga. En este caso el primer proceso que se ejecutará será *InputQuery* que le entregará un objeto del tipo *Query* al proceso *Client* y que, a su vez, entregará un objeto de la clase *Document[]*[1] al proceso *ShowResults*. En este ejemplo se puede apreciar, además, el principal mecanismo que permite compartir la información los diferentes procesos. El proceso anterior enviará un objeto al siguiente, el cual procesará la información y enviará otro objeto al siguiente proceso.

Las ejecuciones podrán contener procesos, métodos y parámetros locales como aparece en el siguiente código:

```
<run name="Execution_1" title="Example_of_first_execution">
  <!-- Local parameters -->
  <parameters>...</parameters>
```

[1]En JAVA los arrays son tomados también como objetos.

```
  <!-- Local methods -->
  <methods>...</methods>
  <!-- Local processes -->
  <processes>...</processes>
</run>
```

Cada ejecución tendrá un identificador único (*name*) y un título explicativo (*title*). El identificador será utilizado por el JPM para discernir qué ejecución será la que se ejecutará cuando se lance la aplicación.

4.3.3. Procesos y subprocesos

Las ejecuciones no pueden ser implementadas por el desarrollador de la aplicación, simplemente se definen en el archivo de configuración. Es el JPM el encargado de lanzar automáticamente los procesos que contenga. Sin embargo, un desarrollador puede implementar nuevos procesos y añadirlos al archivo de configuración para que lleve a cabo una tarea determinada. Los procesos siempre reciben un objeto y devuelven otro como se puede apreciar en la figura 4.4.

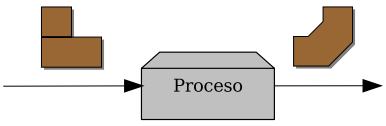


Figura 4.4: Un proceso

Pero no se limitan a esto. Los procesos pueden llamar a otros subprocesos y métodos así como consultar cualquier parámetro del archivo de configuración como se observa en la figura 4.5. Los subprocesos, como los procesos, sólo reciben un objeto y devuelve otro, mientra que los métodos pueden recibir muchos objetos como argumento pero sólo pueden devolver uno.

En el siguiente código se aprecia cómo los procesos, al igual que las ejecuciones, pueden contener otros procesos, métodos y parámetros locales:

```
<process name="process_id" title="Description_of_process"
         class="JAVA_class_to_run">
  <!-- Local parameters -->
```

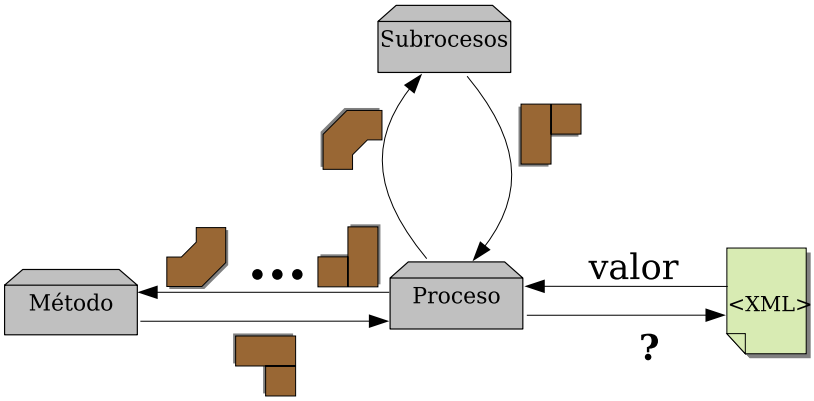


Figura 4.5: Funciones de un proceso

```
  <parameters>...</parameters>
  <!-- Local methods -->
  <methods>...</methods>
  <!-- Local processes -->
  <processes>...</processes>
</process>
```

Los procesos podrán lanzar los procesos que tengan definidos dentro de la etiqueta *processes* pero será labor del desarrollador decidir si el proceso lanzará sus subprocesos y cuando.

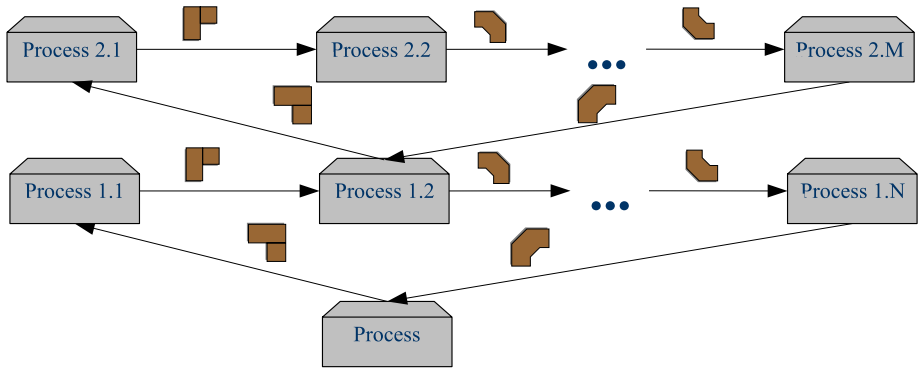


Figura 4.6: Ejemplo de subprocesos

En la figura 4.6 se puede observar un ejemplo de subprocesos. En este ejemplo el proceso *Process* lanza, en un determinado momento, los subprocesos que tiene asignados. Al lanzar los subprocesos, se les entrega un objeto, a cambio, los subprocesos siempre devuelven otro como resultado de sus operaciones. Cada subproceso puede, a su vez, lanzar otros subprocesos entregando siempre un objeto y obteniendo a cambio otro. Si el proceso o subproceso no necesita un objeto de entrada se le puede entregar un objeto vacío. Así mismo, si el proceso no tiene ningún objeto para devolver, éste puede entregar, también, un objeto vacío.

Los procesos pueden ser globales o locales. Los procesos globales son procesos que pueden ser accedidos por cualquier ejecución, proceso o método. Además, son los únicos procesos que pueden ser referenciados a partir de su identificador. Los procesos locales sólo pueden ser invocados por la ejecución, proceso o método que lo contenga. Además, estos procesos no se pueden invocar individualmente, sino que se invocaría junto con el resto de procesos del mismo nivel. En el siguiente código aparecen procesos globales, locales y referencias a procesos globales:

```
<!-- Global processes -->
<processes>
  <process name="process_1" title="Title 1" class="class_1">...</process>
  <process name="process_2" title="Title 2" class="class_2">...</process>
  ...
  <process name="process_n" title="Title n" class="class_n">...</process>
</processes>
<!-- A run -->
<run name="run_1" title="Run title 1">
  <!-- Local processes -->
  <processes>
    <process name="process_1_1" title="Title 1_1" class="class_1_1">
      ...
    </process>
    <!-- Reference to a global process -->
    <process name="process_1_2" title="Title 1_2" class="$process_1" />
    ...
    <process name="process_1_n" title="Title 1_n" class="class_1_n">
      ...
    </process>
  </processes>
</run>
```

Los procesos *process_1*, *process_2*, ..., *process_n* son procesos globales mientras que *process_1_1*, *process_1_2*, ..., *process_1_n* son procesos locales a la ejecución *run_1*. El proceso *process_1_2* en realidad

es una referencia al proceso global *process_1*, pues no contiene una clase en el atributo *class* sino que contiene el identificador del proceso global *process_1* precedido del símbolo $. Así, cuando se invoque al proceso *process_1_2* en realidad se estará ejecutando el proceso global. Cuando el usuario mande ejecutar la ejecución *run_1*, el JPM ejecutará de forma secuencial todos los procesos que contenga este ejecución entre las etiquetas *processes* y */processes*. Si uno de los procesos de *run_1* tuviera, a su vez, otros procesos declarados de la misma forma, este proceso también ejecutaría estos subprocesos de forma secuencial y nunca podría ejecutar uno y el resto no como ocurre con los procesos globales.

La principal ventaja de los procesos locales, con respecto a los procesos globales, ejecuciones o métodos, es que los procesos locales pueden ser ejecutados de forma paralela. JPM se encarga de sincronizar todos los procesos paralelos y de devolver un array con todos los objetos devueltos por estos procesos. Para ejecutar los procesos en paralelo únicamente hay que sustituir las etiquetas *processes* y */processes* por *parallelprocesses* y */parallelprocesses* tal y como aparece en el siguiente código:

```
<run name="run_1" title="Run title 1">
  <!-- Local parallel processes -->
  <parallelprocesses>
    <process name="process_1_1" title="Title 1_1" class="class_1_1">
      ...
    </process>
    <!-- Reference to a global process -->
    <process name="process_1_2" title="Title 1_2" class="$process_1" />
    ...
    <process name="process_1_n" title="Title 1_n" class="class_1_n">
      ...
    </process>
  </parallelprocesses>
</run>
```

De la misma forma, y al igual que ocurría con los procesos que se ejecutaban de forma secuencial, un proceso parelo puede contener otros subprocesos (ya sean secuenciales o paralelos). En la figura 4.7 vemos un ejemplo de procesos paralelos.

Cualquier ejecución, proceso o método puede ejecutar procesos paralelos como aparece en la figura. La acción que llama a los procesos paralelos envía el mismo objeto a cada uno de ellos. Estos se ejecutan recurrentemente y JPM sincroniza la terminación de todos, devolviendo a la acción original un array de objetos con los resultados de cada uno

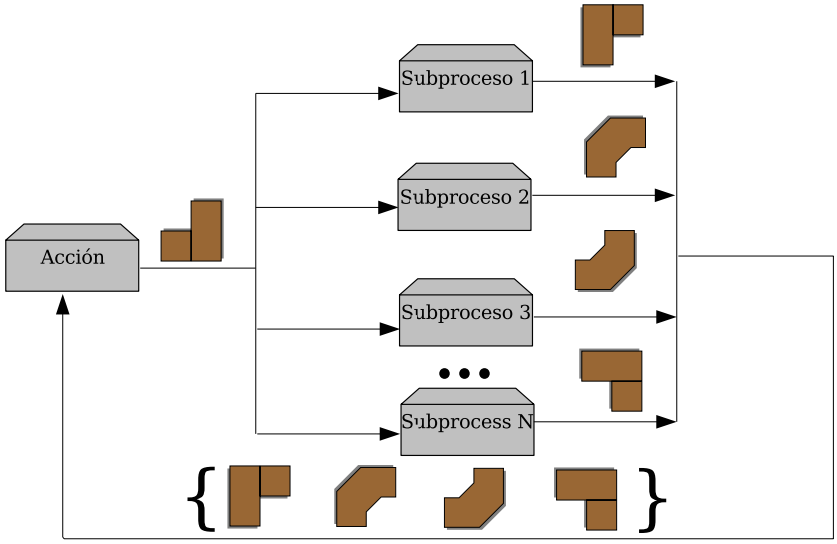


Figura 4.7: Ejemplo de procesos paralelos

de los procesos.

Como veremos más adelante, esta posibilidad de lanzar procesos paralelos nos permitirá usar JPM para realizar aplicaciones distribuidas.

4.3.4. Métodos

Los procesos tienen dos limitaciones: una es que sólo pueden ejecutarse en determinado orden sin repetirse dentro de un mismo nivel y que sólo puede recoger un parámetro y devolver otro. En ocasiones es necesario lanzar alguna tarea que necesite varios parámetros y que no tenga necesidad de ejecutarse en un orden preestablecido. Para ello se definieron los métodos. Como se aprecia en la figura 4.8, los métodos son muy parecidos a los procesos.

Al igual que los procesos, los métodos pueden consultar los parámetros del sitema, llamar a otros métodos e invocar a los subprocesos que tengan definidos. La diferencia más importante es que los métodos pueden recibir varios objetos o ninguno y, además, pueden devolver cualquier tipo de datos, no sólo objetos, así como no devolver tampoco ninguno. Esto da mucha libertad a la hora de crear nuevos métodos, pero el JPM

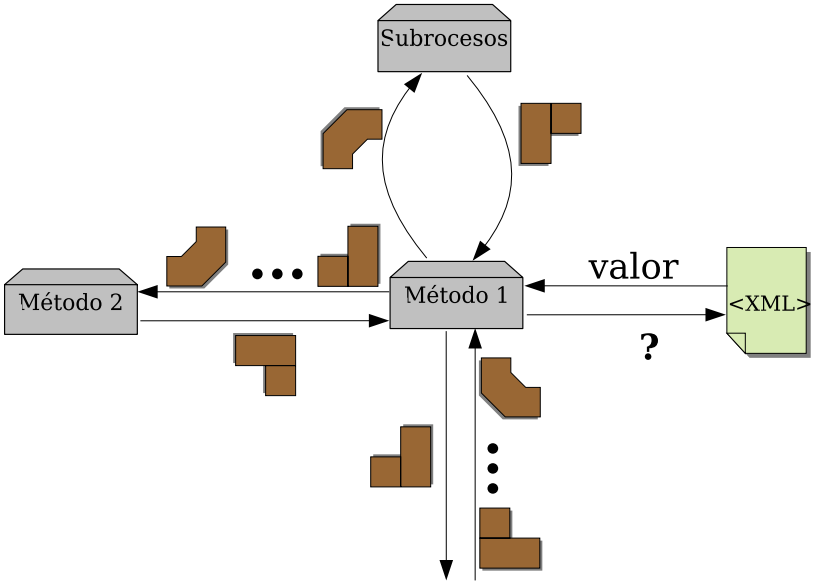


Figura 4.8: Arquitectura de los métodos

no los ejecuta automáticamente como hacía con los procesos ni de forma paralela. Los métodos siempre deben ser invocados expresamente por su identificador donde quieran ser usados. Los métodos también pueden ser globales y locales. Los métodos globales pueden ser accedidos por cualquier proceso o método. Sin embargo, los métodos locales únicamente pueden ser accedidos por todos aquellos procesos o métodos que se encuentren en el mismo nivel, o por debajo. En el siguiente pedazo de código mostramos algunos ejemplos de métodos:

```
<!-- Global method -->
<methods>
  <method name="method_1" title="Title_m1" class="class_m1">...</method>
</methods>
<!-- A run -->
<run name="run_1" title="Run_title_1">
  <!-- Local method -->
  <methods>
    <method name="method_2" title="Title_m2" class="class_m2">...</method>
  </methods>
  <!-- Local process -->
  <processes>
    <process name="process_1" title="Title_p1" class="class_p1">
```

```
      <!-- Local method -->
      <methods>
        <method name="method_3" title="Title_m3" class="class_m3">
        ...
        </method>
      </methods>
      <processes>
        <process name="process_2" title="Title_p2" class="class_p2">
        ...
        </process>
      </processes>
    </process>
  </processes>
</run>
```

El método *method_1* es un método global y puede ser accedido por todos los procesos y métodos que aparecen en el archivo de configuración. También podemos observar dos métodos locales, el *method_2* es local a *run_1* y, por lo tanto, puede ser accedido por *process_1*, *process_2* y *method_3*. Sin embargo el método *method_3* es un proceso local a *process_1* y, por lo tanto, sólo puede ser accedido por *process_1* y *process_2*. En este caso, el método *method_2* no podría llamar a *method_3* al estar éste último en un nivel inferior.

Los métodos son muy útiles para ampliar la configuración de la aplicación. Esto es así porque el usuario no sólo puede modificar los parámetros de la aplicación sino escoger qué métodos se adaptan mejor a sus necesidades. Por ejemplo, imaginemos que tenemos la colección de documentos comprimida en formato *zip*. JIRS es capaz de tratar ficheros comprimidos en varios formatos así que, únicamente, le indicamos que *método* de descompresión debe utilizar con los ficheros de la colección de documentos. Si cambiamos de colección de documentos y, por lo tanto, de formato de compresión, únciamente debemos cambiar, en el archivo de configuración, el método específico de descompresión que debe utilizar y la aplicación funcionaría correctamente con la nueva colección sin tener que modificar el código de la aplicación. En caso de que JIRS no soportara ese formato, un programador podría crear un nuevo método por su cuenta, respetando la interfaz establecida para ese tipo de métodos, y, simplemente, añadirlo en la configuración. Esto tiene varias ventajas: el usuario no tiene por qué conocer cómo está programado JIRS o JPM, únicamente debe crear un nuevo método; aumenta el grado de escalabilidad de JIRS, permitiendo añadir operatividad sin que, por ello, se vea

afectado el resto de la aplicación; y, como se ha explicado anteriormente, aumenta el grado de configuración de la aplicación, al poder especificar nuevos métodos en el archivo de configuración. Además, y como veremos más adelante, JIRS permite que los procesos y métodos puedan ser escogidos dependiendo de los parámetros del sistema. Es decir, que se puede asociar un parámetro a uno o varios procesos o métodos así, dependiendo del valor de este parámetro, JPM puede decidir qué proceso o método ejecutar. De esta forma, podríamos crear un parámetro llamado, por ejemplo, *decompress_ method* que dependiendo de si su valor es *zip* o *gz*, JPM selecionaría el método para descomprimir ficheros *zip* o archivos *gz* respectivamente.

4.4. Parámetros

Los parámetros son pares de nombre-valor que permiten pasar información a las ejecuciones, procesos o métodos. Se utilizan, o bien para definir las diferentes configuraciones del sistema JIRS, o para intercambiar datos globales entre procesos y métodos. Los parámetros pueden ser globales, locales, dinámicos o estáticos. Los parámetros globales son aquellos que pueden ser accedidos por cualquier proceso o método definido en el archivo de configuración y los locales sólo por un subconjunto de éstos. Los parámetros dinámicos son parámetros globales que han sido declarados por algún proceso o método durante la ejecución de la aplicación o mediante la línea de órdenes cuando se invoca al JPM.

4.4.1. Parámetros estáticos

Los parámetros estáticos son aquellos que se definen en el archivo de configuración y pueden ser parámetros globales y locales. Un ejemplo se puede observar en el siguiente código:

```
<!-- Global parameter -->
<parameters>
  <parameter name="param_1">value_1</parameter>
</parameters>
<!-- A run -->
<run name="run_1" title="Run title 1">
  <!-- Local parameter -->
  <parameters>
```

```
    <parameter name="param_2">value_2</parameter>
  </parameters>
  <!-- Local process -->
  <processes>
    <process name="process_1" title="Title_p1" class="class_p1">
      <!-- Local parameter -->
      <parameters>
        <parameter name="param_3">value_3</parameter>
      </parameters>
      <processes>
        <process name="process_2" title="Title_p2" class="class_p2">
          <parameters>
            <parameter name="param_4">value_4</parameter>
          </parameters>
        </process>
      </processes>
    </process>
  </processes>
</run>
```

En este código hay 4 parámetros estáticos, de los cuales *param_1* es global y puede ser accedido por cualquier proceso o método definido dentro de cualquier ejecución. El resto de parámetros son locales aunque con diferente ámbito. El parámetro *param_2* con el valor *value_2* es un parámetro local a *run_1* y su valor puede ser obtenido desde cualquier proceso o método de la ejecución *run_1* pero no desde otras ejecuciones. El parámetro local *param_3* es local al proceso *process_1* y por lo tanto sólo puede ser accedido por este proceso y por el proceso *process_2*. Sin embargo, el parámetro *param_4* sólo puede ser leido por el proceso *process_2* al ser unicamente local a éste.

En el caso de haber dos parámetros, uno local y otro global, con el mismo nombre dentro de un archivo de ejecución, un proceso que pregunte por el parámetro con ese nombre, siempre accederá al que tiene más próximo. Es decir, tendrá preferencia el parámetro local a ese proceso. Si no se encuentra entre los parámetros locales, se utilizará el parámetro local a alguno de los ascendentes del proceso. En el caso de que ese parámetro no se encuentra tampoco entre los ascendentes, entonces, y sólo en este caso, utilizará el parámetro global. Así, siguiendo el ejemplo anterior, si el parámetro *param_4* tuviera el mismo nombre que el parámetro global *param_1*, el proceso *process_2*, al invocarlo, JPM le entregaría el valor *value_4*. Sin embargo, si es el proceso *process_1* el que consulta ese parámetro, JPM le devolverá el valor *value_1* del parámetro global, pues en este proceso ni en sus ascendentes (en este caso la ejecución *run_1*)

no tienen un parámetro local con el mismo nombre.

Los parámetros estáticos se utilizan para definir los parámetros por defecto de la aplicación, como por ejemplo el idioma de la colección de documentos, la dirección TCP/IP del servidor, o del puerto de comunicaciones que utilizará el servidor. Es bastante engorroso ir modificando estos parámetros cuando se quiera ejecutar la aplicación con diferentes parámetros, sobretodo si se está experimentado pues abría que modificar el archivo de configuación cada vez que quisieramos lanzar JIRS con un parámetro distinto. Para resolver este inconveniente se crearon los parámetros dinámicos.

4.4.2. Parámetros dinámicos

Los parámetros dinámicos son parámetros que pueden ser definidos o bien por el usuario o bien por los procesos. Estos parámetros son globales y tienen preferencia sobre cualquier parámetro estático. Así, si definimos un parámetro dinámico con el mismo nombre que un parámetro estático (independientemente de si éste es global o local), cualquier proceso o método que pregunte por él, obtendrá el valor del parámetro dinámico. De esta forma el usuario puede cambiar cualquier parámetro local definiendo un parámetro dinámico como argumento en la línea de órdenes de la aplicación. Para más detalles al respecto se puede leer el apéndice B.

No puede haber dos parámetros dinámicos con el mismo nombre. En el caso de de intentar crear dos parámetros dinámicos con el mismo identificador, entonces el valor del primero será modificado por el valor del segundo parámetro.

Es recomendable que los parámetros dinámicos se usen para cambios esporádicos de los parámetros del sistema. En el caso de querer mantener un parámetro de forma definitiva o por defecto, es recomendable modificarlo directamente en el archivo de configuración.

Otro uso que tienen los parámetros dinámicos es para intercambiar información entre procesos y métodos. Estos parámetros dinámicos definidos por los procesos y métodos permiten resolver las limitaciones que los procesos y métodos tienen con respecto a sus argumentos y a los valores que devuelven. Como hemos visto en los apartados 4.3.3 y 4.3.4 a los procesos sólo se les puede entregar un objeto y éstos sólo devuelven otro; los métodos sólo pueden devolver, como máximo, uno; y los argumen-

tos de los procesos no se pueden *saltar* procesos intermedios, es decir, un proceso sólo puede entregar el objeto devuelto al siguiente proceso y, sino existe, a su antecesor, no puede saltarse esta cadena. Pero como los procesos y métodos pueden definir nuevos parámetros dinámicos en tiempo de ejecución, un proceso o método podría crear tantos parámetros dinámicos con sus valores correspondientes como necesitase para que cualquier otro proceso o método pueda acceder a ellos. Un ejemplo se muestra en la figura 4.9.

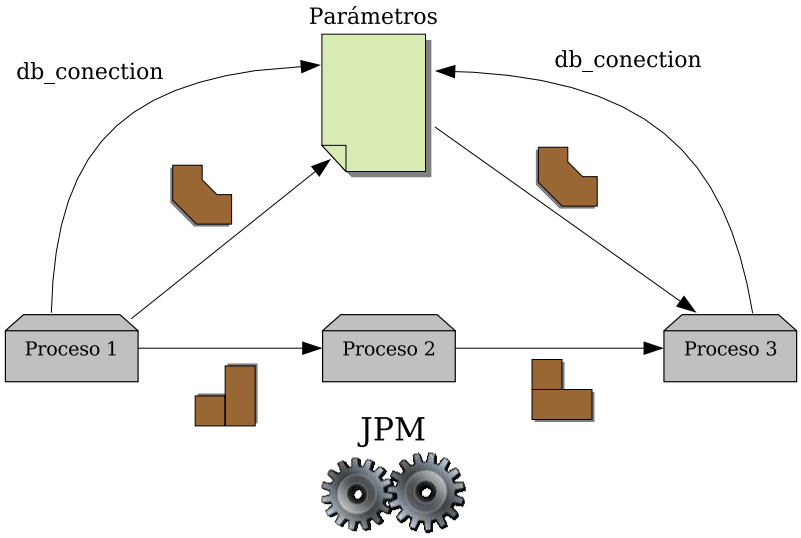


Figura 4.9: Intercambio de información entre dos procesos utilizando parámetros dinámicos

En el ejemplo podemos ver la estructura típica de procesos en la cual un proceso pasa un objeto a otro y éste a su vez al siguiente. La diferencia reside en que el proceso *process_1* define un parámetro dinámico con el nombre *db_connection* y le asigna un valor (en este caso un objeto). Por otra parte, el proceso *process_3* recoge el objeto del parámetro definido en *process_1*. Con este ejemplo, vemos que el proceso *process_3* no sólo recoge el objeto que le entrega el proceso *process_1* sino que, además, puede obtener objetos procedentes de otros procesos o métodos.

4.4.3. Referencia entre parámetros

Una de las ventajas de los parámetros estáticos es que éstos pueden hacer referencias a otros parámetros de tal forma que el valor de un parámetro se forme a partir del valor de otros. Por supuesto, estos parámetros referenciados deben poder se accedidos y no ser locales a ninguna ejecución, proceso o método en los que el parámetro no tenga acceso.

Las referencias entre parámetros se realiza cuando el valor de un parámetro tiene el identificador de otro encerrado entre símbolos $. En el siguiente ejemplo se muestra un ejemplo:

```
<!-- Global parameter -->
<parameters>
  <parameter name="param_1">Hello</parameter>
</parameters>
<!-- A run -->
<run name="run_1" title="Run title 1">
  <!-- Local parameter -->
  <parameters>
    <parameter name="param_2">$param_1$ world</parameter>
  </parameters>
  ...
</run>
```

En este ejemplo se define un parámetro global *param_1* con el texto "*Hello*" y otro parámetro *param_2* local a la ejecución *run_1* cuyo valor está compuesto por el valor del parámetro *param_1* concatenado por el texto " *world*". Así, si cualquier acción interrogase a este parámetro, entonces, su valor sería "*Hello world*". Si se quisiera poner como valor un símbolo $ es necesario poner dos juntos $$, por ejemplo:

```
<!-- Global parameter -->
<parameters>
  <parameter name="param_1">The dollar symbol is $$</parameter>
</parameters>
```

En este caso el valor del parámetro *param_1* sería "*The dollar symbol is $*".

4.4.4. Array de parámetros

Los parámetros estáticos pueden formar lo que se llaman array de parámetros. Es decir, un único parámetro con múltiples valores. Para

formar array de parámetros solamente hay que definir, en un mismo nivel, varios parámetros con el mismo nombre como se muestra en el siguiente ejemplo:

```
<parameters>
  <parameter name="param_1">value_1</parameter>
  <parameter name="param_1">value_2</parameter>
  <parameter name="param_1">value_3</parameter>
</parameters>
```

Con esta configuración una acción que tuviera acceso a este parámetro podría utilizarlo como array. En realidad, cualquier parámetro puede tratarse como array, pues en el caso de que sólo hubiese un parámetro con el mismo nombre en el mismo nivel, JPM lo trataría como un array de un elemento.

4.5. Ámbito de las acciones y parámetros

Como hemos visto en los apartados anteriores, existen tres tipos de acciones que se pueden definir en el archivo de configuración: las ejecuciones, los procesos y los métodos. Las ejecuciones siempre son globales pues se definen en el más alto nivel del archivo de configuración y únicamente pueden ser referenciados desde la línea de órdenes del JPM. Ningún otro proceso o método puede llamar a una ejecución determinada, únicamente lo hace el usuario en el momento de lanzar la aplicación para decidir qué tarea realizará JPM. Por ejemplo, imaginemos que en el archivo de configuración se definen tres ejecuciones: *indexar*, *buscar pasajes* y *ver el estado de la base de datos y ficheros invertidos*. Es tarea del usuario decidir cual de estas tres tareas realizará JPM durante su próxima ejecución. Para más información sobre la línea de órdenes y las tareas que puede realizar JIRS consultar el apéndice B. Por otra parte los procesos y métodos pueden ser globales y locales según dónde se encuentren definidos dentro del archivo de configuración. Si éstos están definidos en el más alto nivel del archivo de configuración serán globales pero si se encuentran dentro de alguna ejecución, proceso o método entonces serán locales a éste,

Por su parte, los parámetros además de ser globales y locales, también pueden ser dinámicos y estáticos. Los parámetros dinámicos, como

hemos mencionado, siempre son globales pero tienen preferencia sobre cualquier parámetro estático, ya sea local o global. En contra posición, los parámetros estáticos locales tienen preferencia sobre los globales, pero éstos sólo pueden ser accedidos por la acción en la que están definidos o por otros procesos y métodos que se encuentren en niveles más bajo de la jerarquía.

También se ha mostrado cómo los procesos globales y los métodos pueden ser accedidos por otros procesos y métodos, en tal caso, estos procesos y métodos adquieren, en tiempo de ejecución, el ámbito del proceso o método invocador. Esto permite que estas acciones puedan acceder a los métodos y parámetros locales al invocador, cosa imposible si mantuvieran su condición de globales.

4.6. JAVA Database Information Retrieval

Entre las clases de RP se incluyen las JAVA Database Information Retrieval (JDBIR) que es un pequeño gestor de bases de datos escrito también en JAVA desarrollado específicamente para sistemas RI pues soporta diversos tipos de listas invertidas e índices.

Siguiendo la línea de JIRS, los archivos que definen cómo tendrá que ser la base de datos son archivos XML. La definición es muy intuitiva y su estructura general se puede ver a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<database name="database_id" path="path_to_directory">
  <tables>
    <table name="table_name_1">
      ...
    </table>
    <table name="table_name_2">
      ...
    </table>
    ...
    <table name="table_name_N">
      ...
    </table>
  </tables>
</database>
```

Así, una base de datos tendrá un nombre que servirá de identificador para futuras referencias y un directorio dónde se almacenará los ficheros

de la base de datos así como los ficheros invertidos. Cada base de datos tendrá una o más tablas con sus respectivos nombres que también servirán como identificadores.

Cada tabla se define de la siguiente forma:

```
<table name="table_name">
  <fields>
    <field name="field_name_1" type="field_type" atributes... />
  </fields>
</table>
```

Cada table se compone de uno o más campos. Aunque cada tipo de campo puede tener diferentes atributos, todos los parámetros tiene dos parámetros comunes: nombre del campo y el tipo. Aunque se puede extender fácilmente los tipos en futuras versiones del gestor de base de datos, los tipos que actualmente se soportan son los siguientes:

char Los campos de tipo *char* permiten almacenar cadena de caracteres de una longitud máxima. Estos campos tienen los siguientes atributos:

- **length** Longitud máxima de la cadena de caracters que se pueden almacenar en este campo.
- **encoding** Codificación de caracteres que se utilizará para almacenar los datos.

int Este tipo de campos se utiliza para almacenar números enteros de 32 bits.

float Para campos que almacenen números de coma flotante de 32 bits.

vchar Este tipo son para los campos que soportan ficheros invertidos con cadena de caracteres.

vint Estos campos se utilizan para crear listas variables de enteros sobre ficheros invertidos.

vfloat Campos que referencian aFicheros invertidos con números de coma flotante.

También se pueden definir varios tipos de claves primarias para realizar búsquedas muy eficientes y un tipo de clave ajena para referenciar a campos de difentes tablas. La única limitación de las claves primarias y ajenas es que no pueden ser incorporadas en campos de ficheros invertidos, únciamente a los campos de tipo *char*, *int* y *float*.

Así, cada ampo puede tener una clave primeria y una clave ajena. Las claves primarias son de dos tipos:

index Este tipo de clave primaria se utiliza para búsquedas por el valor del campo. Internamente se construye una tabla hash en memoria por cada campo con este tipo de claves primarias en su definición. Esto permite encontrar un registro de forma casi inmediata sin realizar un sólo acceso a disco.

array Cuando lo que se quiere obtener es el valor de un campo de forma rápida, lo que se crea es una clave primaria de tipo array sobre este campo, lo que permite encontrar su valor de forma casi inmediata pues JDBIR construye un array en memoria del mismo tamaño que el número de registros de la tabla y almacena en cada elemento del array el valor del campo en cada registro de la tabla.

Las claves primarias se definen mediante el atributo *key* en la definición del campo, especificando en ese momento el tipo: *index* para las del tipo índice y *array* para las del tipo array.

Las claves ajenas solo tienen un tipo y hacen referencia a un campo de una tabla. Esto indica que el valor del campo que contiene la tabla ajena hace referencia a un registro de otra tabla. El atributo para definir claves ajenas es *ref* y se debe colocar el nombre de la tabla seguido de un punto y del nombre del campo al que hace referencia. Este sería la configuración para la base de datos de JIRS:

```
<?xml version="1.0" encoding="UTF-8"?>
<database name="jirsdb" path="$database_dir$">
  <tables>
    <table name="Documents">
      <fields>
        <field name="docno" type="char" length="30" encoding="UTF-8" />
        <field name="location" type="int" ref="DocumentsLoc.location" />
      </fields>
    </table>
    <table name="DocumentsLoc">
```

```
      <fields>
        <field name="location" type="vchar" encoding="UTF-8" />
      </fields>
    </table>
    <table name="Passages">
      <fields>
        <field name="iddoc" type="int" ref="Document.docno" key="array"/>
        <field name="weight" type="float" key="array"/>
        <field name="freqmax" type="int" key="array"/>
        <field name="text" type="vchar" enconding="UTF-8" />
      </fields>
    </table>
    <table name="TermsDocument">
      <fields>
        <field name="terms" type="vint" />
        <field name="freqs" type="vint" />
      </fields>
    </table>
    <table name="Histogram">
      <fields>
        <field name="idterm" type="int" />
        <field name="word" type="char" length="20" encoding="UTF-8"
               key="index" />
        <field name="freq" type="int" />
        <field name="ndocs" type="int" />
        <field name="docs" type="vint" ref="Passages.iddoc" />
        <field name="freqs" type="vint" />
      </fields>
    </table>
  </tables>
</database>
```

Con esto se construye una base de datos cuyo nombre es *jirsdb* y cuyos ficheros y ficheros invertidos se almacenarán en el directorio que está definido en el parámetro *database_dir*. Al igual que ocurría con el archivo principal de configuración del JPM, en la base de datos también se puede hacer referencias a los parámetros del sistema. En este caso esperamos que el directorio donde se almacenará la base de datos ya éste establecido en el archivo de configuración del JPM.

Esta base de datos contiene 5 tablas. En la primera tabla (*Documents*) se almacena el identificador del documento que se quiere indexar y su localización física real, es decir, la ruta al archivo que contiene ese documento. Como JIRS soporta que un mismo archivo contenga varios documentos como hacen los congresos internacionales del CLEF y del TREC, y para no duplicar la información, el campo *location* es en realidad una clave ajena a la tabla de *DocumentLoc* que contiene todas las posiciones físicas de los documentos de la colección. La tercera tabla es

Passages que contiene la información de cada pasaje del sistema como el identificador del documento al que pertenece, el peso del pasaje, la frecuencia máxima de los términos y el texto. Tanto el peso del pasaje como la frecuencia máxima se utilizarán para calcular la función *tf/idf* del modelo espacio vectorial. La tabla *TermsDocument* almacenará los identificadores y las frecuencias de los términos que un documento dado contiene. Por último, la tabla *Histogram* almacenará información relevante de cada término, como su identificador, la misma palabra, su frecuencia en la colección documentos, el número de documentos en los que aparece, la lista de documentos que contienen esa palabra y la frecuencia de cada aparecición en cada documento.

4.7. Acciones y parámetros condicionados

El JPM permite modificar los parámetros del sistema de forma dinámica mediante el uso de los parámetros dinámicos. Esto evita que, cada vez que queramos cambiar algún parámetro del sistema, tengamos que editar el archivo de configuración. Por desgracia, éste método sería muy difícil de utilizar si se aplicara también a los procesos y métodos de una ejecución. Es decir, modificar dinámicamente el comportamiento de la aplicación sin tener que editar el archivo de configuración de la misma forma que se modifica los parámetros. También las referencias entre parámetros nos permiten modificar el valor de un parámetro dependiendo del valor de otro. Esto limita el valor final de los parámetros pues siempre contendrá el valor de los parámetros a los que hace referencia. Así, si quisieramos tener un parámetor cuyo valor dependiera del valor de otro parámetro pero que fuera radicalmente distinto, no lo podríamos hacer. Para evitar estos inconvenientes se desarrolló las acciones y parámetros condiconales. Es decir, que un parámetro tenga un valor u otro, y una acción ejecute tal o cual proceso o método y que esto dependa del valor de otro parámetro. Esto permite que se pueda cambiar no sólo la configuración del sistema sino también su comportamiento cambiando unos pocos parámetros. Lo que permite las acciones y parámetros condicionados es la etiqueta *switch* cuya sintaxis es:

```
<switch param="parameter_name">
  <case value="value_1">
```

```
    <!-- Action o parameter defintions -->
  </case>
  <case value="value_2">
    <!-- Action o parameter defintions -->
  </case>
  ...
  <case value="value_n">
    <!-- Action o parameter defintions -->
  </case>
  <otherwise>
    <!-- Action o parameter defintions -->
  </otherwise>
</switch>
```

Dependiendo del valor del parámetro *parameter_ name*, se definirán los parámetros y acciones o bien del primer *case*, del segundo, del tercero, etc. En el caso de que el parámetro no tenga ninguno de los valores definidos en ninguno de los atributos *value* del las etiquetas *case*, entonces se definirán los parámetros y acciones de la etiqueta *otherwise*. La etiqueta *otherwise* es opcional y, en el caso de no estar presente, y el parámetro no coincida con ninguna de los valores anterioes, el JPM no realizará ninguna acción. En el siguiente código vemos un ejemplo de la configuración de parámetros, métodos y procesos condicionales:

```
<!-- Global parameter -->
<parameters>
  <parameter name="param_1">value_1_1</parameter>
</parameters>
<!-- Ejecution -->
<run name="run_1" title="Title_1">
  <!-- Local parameters -->
  <parameters>
    <!-- Conditional parameters -->
    <switch param="param_1">
      <case value="value_1_1">
        <parameter name="param_2">value_2_1</parameter>
      </case>
      <case value="value_1_2">
        <parameter name="param_2">value_2_2</parameter>
      </case>
      <otherwise>
        <parameter name="param_2">value_2_3</parameter>
      </otherwise>
    </switch>
  </parameters>
  <methods>
    <!-- Conditional methods -->
    <switch param="param_1">
      <case value="value_1_1">
        <method name="method_1" title="Title_1" class="class_m1" />
```

```
      </case>
      <case value="value_1_2">
        <method name="method_1" title="Title 1" class="class_m2" />
      </case>
      <otherwise>
        <method name="method_1" title="Title 1" class="class_m3" />
      </otherwise>
    </switch>
  </methods>
  <processes>
    <process name="process_1" title="Title p1" class="class_p1" />
    <!-- Conditional processes -->
    <switch param="param_1">
      <case value="value_1_1">
        <process name="process_2" title="Title p2" class="class_p2" />
      </case>
      <case value="value_1_2">
        <process name="process_3" title="Title p3" class="class_p3" />
      </case>
      <otherwise>
        <process name="process_4" title="Title p1" class="class_p4" />
      </otherwise>
    </switch>
    <process name="process_5" title="Title p1" class="class_p5" />
  </processes>
</switch>
```

En este ejemplo tanto la configuración y el comportamiento de la ejecución *run_1* está condicionados al valor del parámetro *param_1*, pues si este valor es *value_1_1*, entonces el parámetro *param_2* tendrá el valor *2_1*, si se invoca al método *method_1* se invocará a la clase *class_m1* y la secuencia de procesos a ejecutar será *process_1*, *process_2* y *process_5*. Sin embargo, si el valor del parámetro *param_1* es *value_1_2* entonces el valor del parámetro *param_2* será *value_2_2*, el método *method_1* se ejecutará con la clase *class_m2* y la secuencia de procesos será *process_1*, *process_3* y *process_5*. Y si el valor del parámetro *param_1* no es igual al ninguno de los anteriores entonces se declararán o se ejecutarán los parámetros, métodos y procesos encerrados entre etiquetas *otherwise* y no los otros casos.

Con este sencillo caso vemos la potencia que los parámetros, métodos y procesos condiconales tienen a la hora de configurar la aplicación y de determinar su comportamiento modificando unos pocos parámetros.

4.8. JAVA Information Retrieval System

Hasta el momento hemos visto las herramientas que utiliza JIRS para llevar a cabo su trabajo. Estas herramientas han sido desarrolladas para crear un sistema de RI lo más flexible y pontente posible, con gran capacidad de escalabilidad y modificación por terceras personas. De esta forma podemos decir que JIRS es el sistema abierto de RI que permite una mayor adaptabilidad en proyectos de investigación relacionadas con la RI.

JIRS incluye, entre sus posibilidades ya implantadas, la indexación de documentos, la búsqueda de pasajes y de documentos monolingues y multilingües, la evaluación de la precisión y *recall* en tareas clásicas de RI, la evaluación de la cobertura en tareas más relacionadas con RP, aplicar modelos de n-gramas a la búsqueda de RP de distintos sistemas y creación de tablas de cobertura para su futuros análisis.

4.8.1. Indexación de la colección de documentos

Para utilizar JIRS como sistema de RP el primer paso es indexar la colección de documentos, es decir, preparar los datos de forma que, al realizar cualquier búsqueda, ésta sea eficiente y pueda ser obtenida en un tiempo aceptable. Para ello se debe almacenar información sobre documentos, pasajes y términos así como pre-calcular algunos pesos. La arquitectura que adquiere JIRS para indexar se muestra en la figura 4.10.

Las colecciones de documentos del CLEF y del TREC se almacenan en archivos de aproximadamente 1MB. En cada archivo están almacenados los suficientes documentos para alcanzar dicho tamaño. A partir de estos ficheros JIRS recoge cada uno de los documentos almacenando cierta información sobre ellos. Después divide estos documentos en pasajes de 1 frase. Cada pasaje se almacena en un archivo junto con información sobre de qué documento procede esta frase, el término con mayor frecuencia y qué términos relevantes contiene y con qué frecuencia aparecen en dicho pasaje. Los pasajes se dividen en sus elementos constituyentes, términos, y se eliminan las palabras comunes. Después se almacenan información sobre los términos restantes y se construyen los ficheros invertidos necesarios. JIRS almacena el total de veces que cada término es usado en la colección de documentos, el número de pasajes que lo referencia y crea

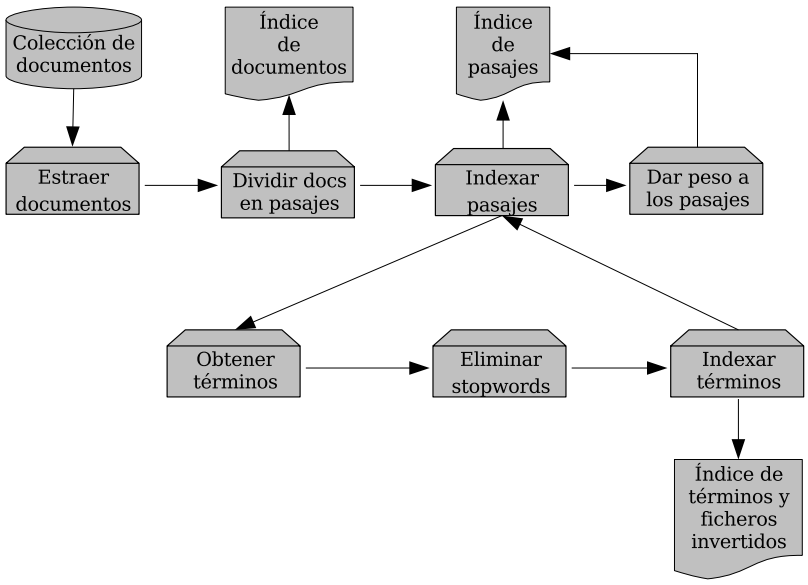


Figura 4.10: Pasos realizados para indexar la colección de documentos

las listas de pasajes y frecuencias en donde ese término aparece. Para terminar, calcula los pesos de los pasajes según la fórmula estándar del modelo espacio vectorial para normailzar los tamaños de éstos durante la búsqueda.

4.8.2. Búsqueda de pasajes local

Una vez indexada la colección de documentos se puede realizar búsquedas de forma eficiente. JIRS puede realizar búsquedas localmente, donde los índices se encuentran en el ordenador del usuario; remotamente, mediante una arquitectura cliente/servidora donde los índices se encuentran en un servidor; y de forma distribuida, donde el trabajo de búsqueda puede ser dividido en diversos servidores. La arquitectura para la búsqueda local se puede apreciar en la figura 4.11.

Las tareas más importantes de JIRS tienen dos modos de visualizar la información: texto y gráfico. Con esto se consigue que JIRS funcione tanto en entornos gráficos como sin ellos (por ejemplo, en servidores

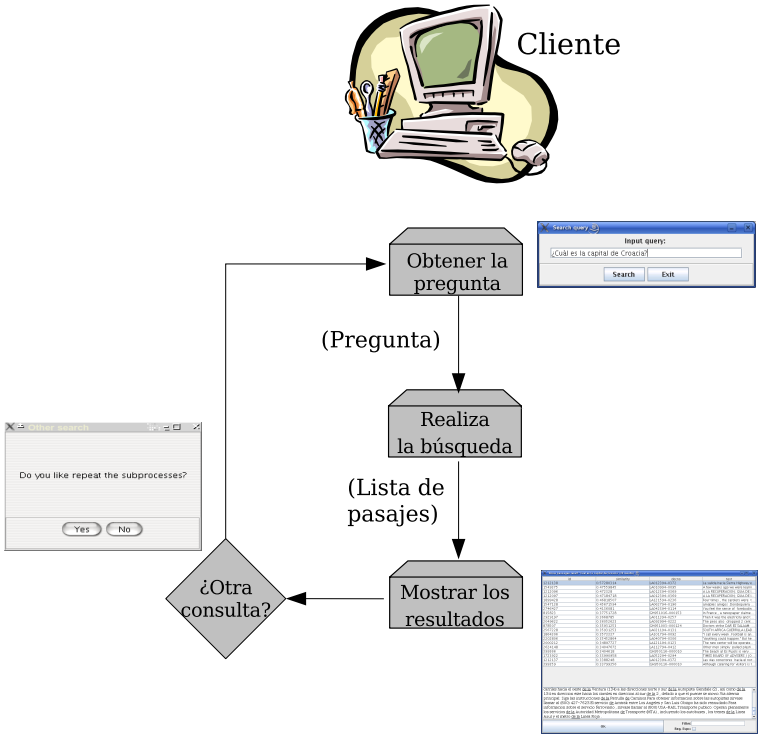


Figura 4.11: El sistema de búsqueda local

Unix). La figura muestra el ejemplo imágenes cuando la aplicación se ejecuta en modo gráfico, aunque la arquitectura en si no cambia por ejecutar JIRS en modo texto. Así, JIRS lo que primero hace es esperar a que el usuario introduzca la pregunta, después se realiza la búsqueda mediante alguno de los métodos implementados en JIRS y, finalmente, se muestran los resultados. JIRS tiene varios métodos de búsqueda: el tradicional modelo espacio vectorial, el RW-Density y los modelos de n-gramas. Sobre los modelos RW-Density y los modelos de n-gramas se explican con más detalle en la sección 3.

4.8.3. Búsqueda de pasajes remota

Gracias al JPM que es parte fundamental del núcleo de JIRS, es muy sencillo adaptar nuestra herramienta local de RP por otra herramienta

que trabaje con arquitectura cliente/servidora. En la figura 4.12 se puede ver un ejemplo.

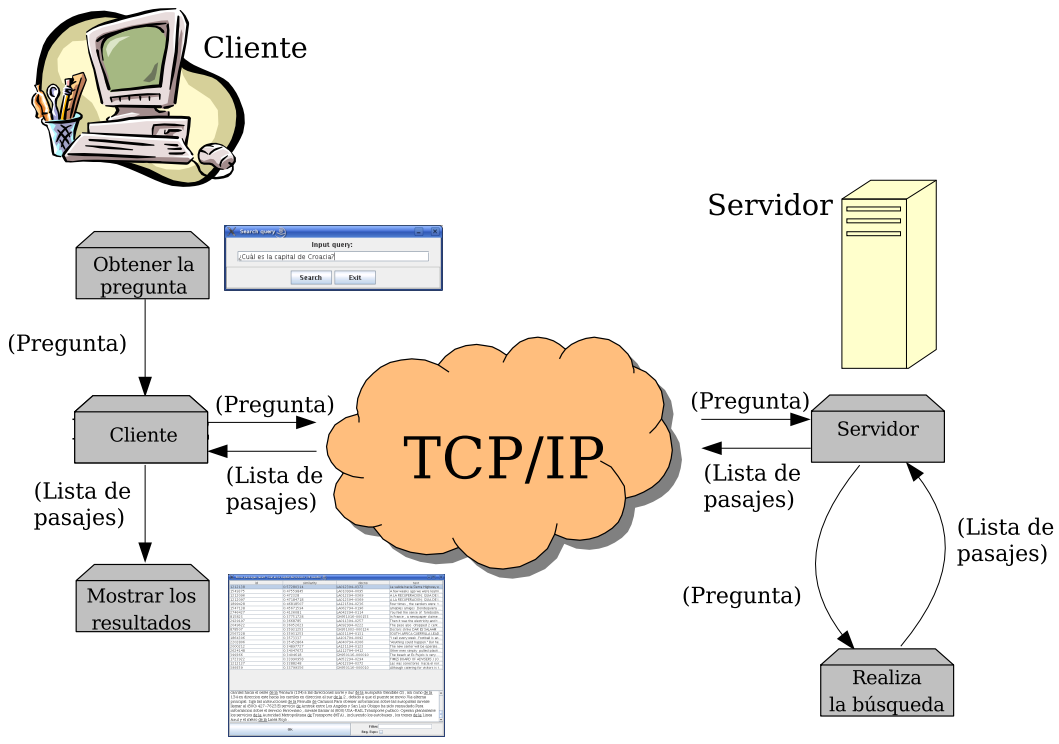


Figura 4.12: El sistema de búsqueda con arquitectura cliente/servidor

JPM aporta dos procesos especiales que permiten el traspaso de información desde un cliente a un servidor. El proceso *cliente* se encarga de recoger el objeto de entrada al proceso y enviarlo mediante TCP/IP a un servidor. Después, este proceso recoge el resultado del servidor y lo pasa al siguiente proceso del archivo de configuración. El proceso *servidor*, por su parte, espera peticiones de cualquier cliente y, cuando recibe un objeto, lo entrega a los subprocesos que tenga definidos en el archivo de configuración. Cuando estos procesos terminan, entregan el resultado al proceso *servidor* que, sin modificarlo, lo envía de nuevo para el cliente. Así, y siguiendo el ejemplo de la figura, el cliente, después de recibir una pregunta por parte del usuario la envía sin modificar al servidor. Éste, a su vez, recoge esa pregunta y la envía a los procesos hijos, que en este

caso es el motor de búsqueda. Una vez que el motor de búsqueda obtiene la lista de pasajes relevantes, se la entrega al proceso *servidor* que enviará dicha lista al proceso *cliente*. Éste estaba esperando el resultado del proceso servidor y, cuando obtiene la lista de documentos la envía al proceso *mostrar los resultados*, que se encargará de presentar los resultados al usuario en modo texto o gráfico según haya escogido el usuario.

4.8.4. Búsqueda de pasajes distribuida

Otra posibilidad de JIRS es la de realizar búsquedas de pasajes de forma distribuida, de tal forma que una misma pregunta pueda ser respondida por multiples sistemas lanzados en múltiples máquinas. Esto se consigue gracias a dos características de JPM: el paralelismo de procesos y la arquitectura cliente/servidora. En la figura 4.13 se puede observar un ejemplo.

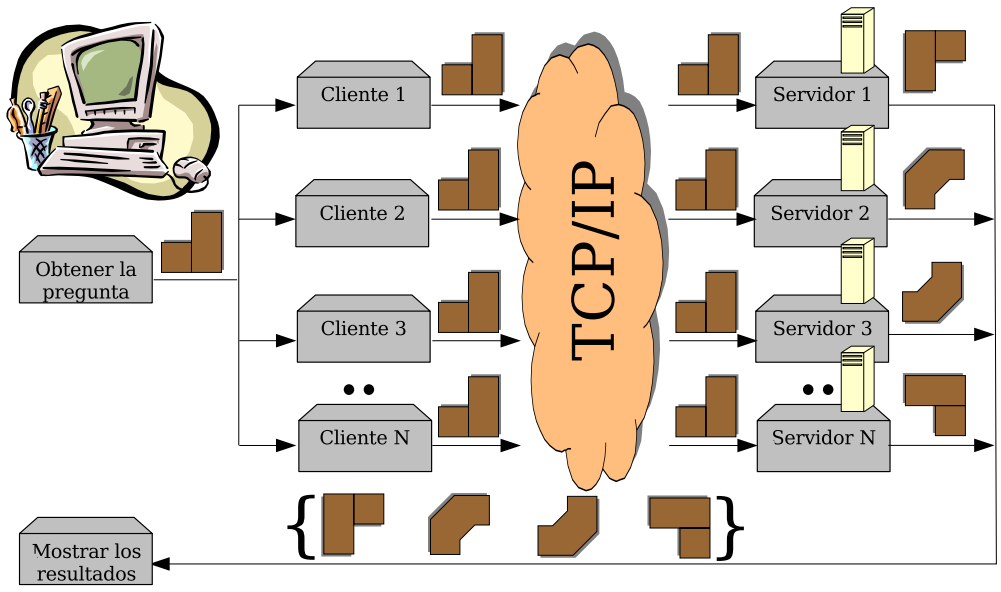


Figura 4.13: El sistema de búsqueda con arquitectura distribuida

En este ejemplo, la pregunta se obtiene de la misma forma que con la búsqueda remota pero con la salvedad de que ésta se envía no sólo a

un cliente sino a varios clientes configurados para que se ejecuten paralelamente. Estos procesos *cliente* enviarán la pregunta a sus respectivos servidores, los cuales las procesarán y enviarán su resultado de vuelta al cliente. JPM sincronizará todas las salidas de los proceso clientes y devolverá los resultados de cada servidor en un único array.

Esta arquitectura es muy útil cuando se aplica a búsquedas multilingües en las cuales la pregunta se hace en un idioma y se buscan en colecciones de documentos de varios idiomas. Así, como en el ejemplo de la figura 4.14, cada servidor podría traducir la pregunta del usuario al idioma de la colección de documentos, buscar la respuesta y, finalmente, volver traducir la respuesta al idioma en que se hizo la pregunta.

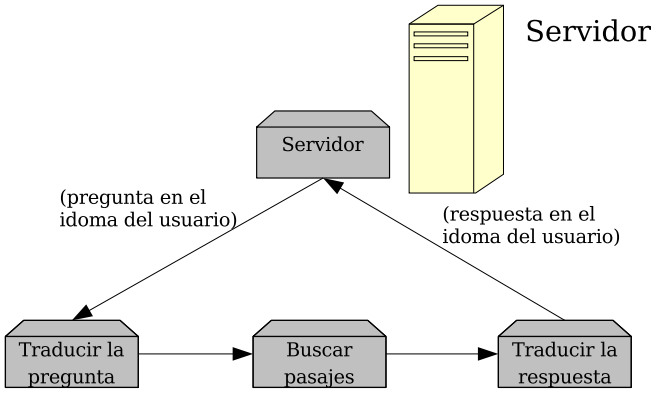


Figura 4.14: Ejemplo de una búsqueda multilingüe

4.8.5. Recuperación de pasajes sobre colección de preguntas

Como se ha comentado, JIRS es una herramienta pensada para la investigación en tareas de RI. Por ello, aporta formas de evaluación automática que permiten, a partir de una colección de preguntas, obtener tablas de coberturas que ayuden a la evaluación de los distintos sistemas de RI. Para ello, como primer paso, es recomendable obtener un archivo con los pasajes que devuelven cada una de las preguntas de la colección. En la figura 4.15 se muestra cómo JIRS lleva a cabo dicha tarea.

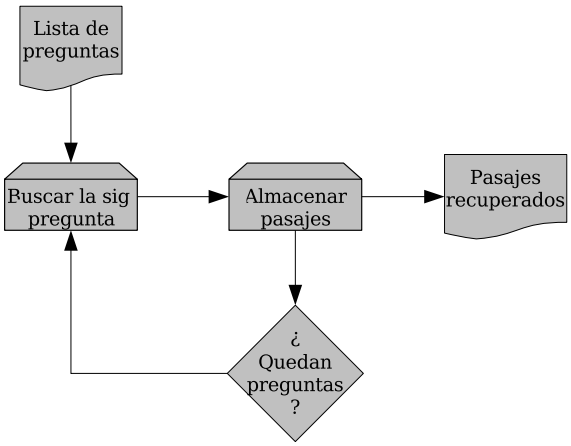


Figura 4.15: Pasos para la recuperación de pasajes sobre colección de preguntas

JIRS, a partir de un archivo una colección preguntas en formato CLEF o XML, busca cada una de las preguntas y guarda los pasajes obtenidos en un archivo XML asociando, por supuesto, cada pasaje con la pregunta con el que se obtuvo. Esto permite analizar posteriomente si estos pasajes son adecuados o no para la tarea de RI que se está evaluando. Esto permite lanzar sucesivamente JIRS con diferentes parámetros y modelos y obtener archivos con los pasajes obtenidos en cada experimento con el objetivo de realizar comparaciones entre ellos.

4.8.6. Creación de tablas de cobertura

Aunque JIRS permite obtener tablas de precisión y *recall* para evaluar el rendimiento de los sistemas, cuando éstos se aplican a la BR es más interesante obtener tablas de cobertura y redundancia. Estas tablas nos dirán en qué pasajes se encuentra la respuesta correcta a partir del archivo de pasajes obtenido en el apartado anterior y un conjunto de expresiones regulares que nos confirmen o no que la respuesta está presente en los pasajes.

Las tablas de cobertura generadas por JIRS son muy sencillas de procesar y obtener otro tipo de datos tales como la redundancia, la precisión y el MRR de cada sistema.

4.8.7. Aplicar modelos de n-gramas a otros sistemas

Como se ha visto en el apartado 3, JIRS implementa un sistema de RP basado en distintos modelos de n-gramas. Estos modelos se pueden aplicar a otros sistemas de RP que no están soportados por JIRS de una forma sencilla a partir de los resultados de éstos.

Capítulo 5

Evaluación de JIRS

En este capítulo se presentan los experimentos realizados con JIRS para: encontrar la configuración óptima de los modelos de n-gramas; comparar dichos modelos con otros sistemas de RP utilizados en la bibliografía; y medir la mejora de precisión que aporta JIRS a sistemas de BR en tareas monolingües y bilingües.

Así, este capítulo empieza con la descripción de las unidades de medida utilizadas para la evaluación de los modelos y sistemas. Seguidamente se describen los corpora usados en las distintas evaluaciones: las colecciones de documentos que se han indexado, los conjuntos de preguntas utilizados en los experimentos y los conjuntos de respuestas creados para evaluar los resultados de las búsquedas. A continuación se han lanzado una serie de experimentos con el objetivo final de encontrar los mejores ajustes o parámetros para los distintos modelos de n-gramas. Esto nos ha servido para enfrentar nuestros mejores modelos con otros sistemas de RP. Estos sistemas han sido ampliamente utilizados en los concursos internacionales del CLEF y del TREC en tareas de BR mono y multilingües. A continuación se ha comprobado el rendimiento de JIRS cuando forma parte de un sistema de BR completo. Además, se han confrontado estos resultados con los obtenidos por el mismo sistema de BR pero utilizando el Lucene[1] como módulo de RP.

[1] http://lucene.apache.org/

5.1. Medidas de evaluación

Para evaluar los experimentos se han utilizado diferentes métricas que nos indicarán en qué grado mejora o empeora el rendimiento del sistema de RP dependiendo de sus parámetros. Ésto también nos permitirá comparar con otros sistemas y modelos existentes. En los siguientes subapartados se describen detalladamente las distintas medidas utilizadas.

5.1.1. Cobertura

Se establece Q como el conjunto de preguntas, P la colección de pasajes, $A_{P,q}$ el subconjunto de P que contiene las respuestas correctas para $q \in Q$, y $R_{P,q,n}$ es el conjunto de n primeros pasajes recuperados de P por el sistema de RP dada una cierta pregunta q.

La cobertura del sistema para una colección de preguntas Q y una colección de pasajes P en una lista ordenada por relevancia de n pasajes es definida por:

$$coverage(Q, P, n) \equiv \frac{|\{q \in Q | R_{P,q,n} \cap A_{P,q} \neq \emptyset\}|}{|Q|} \tag{5.1}$$

Así, la cobertura da la proporción de la colección de preguntas para el cual se puede encontrar una respuesta correcta dentro de los primeros n pasajes recuperados para cada pregunta. Para más detalles véase [RG04].

5.1.2. Mean Reciprocal Rank

Dado el conjunto Q de preguntas, el conjunto P con la colección de pasajes, el subconjunto $A_{P,q}$ perteneciente a P que contiene las respuestas correctas para $q \in Q$, y el conjunto $R_{P,q,n}$ de los n primeros documentos de P devueltos por el motor de búsqueda para una cierta pregunta q, de forma que $R_{P,q,n} = \{p_{q,1}, p_{q,2}, ..., p_{q,n}\}$, se define el *Mean Reciprocal Rank* (MRR) [Voo99] como:

$$mrr(Q, P, n) = \frac{\sum\limits_{\forall q \in Q} rr(R_{P,q,n})}{|Q|} \tag{5.2}$$

donde $rr(R_{P,q,n})$ es una función llamada *Reciprocal Rank* (RR) que depende de la posición del primer pasaje devuelto de la lista de resultados

que contenga la respuesta o 0 si no ha respondido correctamente a la pregunta en los primeros n pasajes. Esta función se define como:

$$rr(R_{P,q,n}) = \begin{cases} \frac{1}{i} & \text{si } \exists i | i = \min\limits_{1 \leq j \leq n} j | p_{q,j} \in A_{p,q} \\ 0 & \text{en otro caso} \end{cases} \tag{5.3}$$

Es decir, el MRR es la media de los RR de cada pregunta. El MRR tiene varias ventajas como unidad de medida para evaluar sistemas de BR. Está muy estrechamente relacionada con la medida de precisión utilizada en RI. Sus límites se encuentran entre 0 y 1 y se promedia bien. Una ejecución es penalizada por no recuperar ninguna respuesta para una pregunta, pero no excesivamente. Sin embargo, la medida tiene algunos inconvenientes. Su valor para una única pregunta puede tener sólo $n+1$ valores (en el caso de una medida de 5-MRR: 0, 0.2, 0.25, 0.33, 0.5 y 1). Esta medida no está adaptada para los sistemas de BR en los que se permite una selección múltiple como respuesta en vez de una única contestación.

5.1.3. Redundancia

Muchos sistemas de BR se basan en la *redundancia* de la respuesta, es decir, cuántas veces se puede encontrar una posible respuesta a una pregunta en los primeros n pasajes para obtener la respuesta más probable. Así, es interesante poder medir la redundancia de un sistema a la hora de incluirlo en un sistema de BR basado en redundancia. La redundancia de las respuestas, ante una colección de preguntas, [RG04] la define como:

$$redundancy(Q, P, n) \equiv \frac{\sum\limits_{q \in Q} |R_{P,q,n} \cap A_{P,q}|}{|Q|} \tag{5.4}$$

Donde Q es el conjunto de preguntas a evaluar, P es la colección de pasajes, $A_{P,q}$ es el conjunto de pasajes de P que contiene una respuesta a la pregunta $q \in Q$, y $R_{P,q,n}$ es el conjunto de los primeros n pasajes de P devueltos por el motor de búsqueda dada la pregunta q.

Así, esta medida nos da la media de las redundancia de los primeros n pasajes devueltos por el motor de búsqueda para cada pregunta.

5.1.4. Precisión

La precisión de un sistema, para un conjunto de preguntas Q y una colección de pasajes P en un rango n de pasajes devueltos ($R_{P,q,n}$), es la proporción de pasajes que contienen la respuesta correcta con respecto al número total de pasajes devueltos. Esta proporción se determina mediante la siguiente ecuación:

$$precision(Q,P,n) \equiv \frac{\sum_{q \in Q} \frac{|R_{P,q,n} \cap A_{P,q}|}{|R_{P,q,n}|}}{|Q|} \tag{5.5}$$

Donde $A_{P,q}$ es el conjunto de pasajes con la respuesta correcta para $q \in Q$.

5.2. El corpus

Los experimentos se han llevado a cabo usando los corpora del CLEF, concretamente los corpora de español, francés, inglés e italiano. En la tabla 5.1 se puede ver un resumen de dichos corpora.

Las colección de mayor tamaño es la formada por la *Agencia EFE* de la colección en español con un 1GB de datos, seguida del inglés con casi 530MB. Las colecciones más pequeñas son la francesa con 438MB y la italiana con, únicamente, 322MB.

Las preguntas realizadas para las evaluaciones son las 200 preguntas que aporta el CLEF para cada una de las tareas monolingües de español, francés e italiano y de las tareas bilingües de español-inglés e inglés-español. En estas 200 preguntas se incluyen preguntas del tipo NIL cuya respuesta no está soportada por la colección de documentos. CLEF distingue las preguntas en 3 tipos:

DEFINITION (D) Son preguntas que piden la definición o cargo de personas, organizaciones o siglas, por ejemplo, "*¿Quién es Javier Solana?*" o "*¿Qué es BMW?*".

TEMPORAL (T) Preguntas que contienen restricciones temporales como por ejemplo "*¿Qué iglesia ordenó mujeres sacerdote en marzo de 1994?*".

Colección	Añadido en	Tam. (MB)	Nº de Docs.	Tam. Medio de Docs. (Bytes)	Tam. Medio de Docs. (Términos)
			Español		
EFE 94	2001	469	215.738	2.281	337
EFE 95	2003	531	238.307	2.336	345
			Francés		
Le Monde 94	2000	143	44.013	3.417	510
Le Monde 95	2006	141	47.646	3.108	472
ATS 94	2001	76	43.178	1.841	277
ATS 95	2003	78	42.615	1.910	288
			Inglés		
LA Times 94	2000	387	113.005	3.587	576
Glasgow Herald 95	2003	141	56.472	2.626	439
			Italiano		
La Stampa 94	2000	176	58.051	3.179	482
AGZ 94	2001	73	50.527	1.522	228
AGZ 95	2003	73	48.980	1.561	235

Tabla 5.1: Resumen de las características de las colecciones de documentos del CLEF

FACTUAL (F) Preguntas que exigen una contestación a un echo puntual, por ejemplo, "*¿A cuánto asciende el premio para la ganadora de Wimbledon?*" o "*¿Cómo murió Jimi Hendrix?*".

En la tabla 5.2 se muestra un resumen de las colecciones de preguntas del CLEF 2005.

5.2.1. Clasificación de las preguntas

El problema de la clasificación de las preguntas del CLEF es que es muy genérica y, para aspectos prácticos, poco útil. Por tanto, nosotros hemos agrupado las preguntas del CLEF en estas 21 categorías:

NAME Esta es una categoría general para nombres y se la asigna a preguntas que se detecta que es una pregunta sobre un nombre

Tarea	#D	#T	#F	#NIL
Monolingüe español	50	118	32	20
Monolingüe francés	50	120	30	20
Monolingüe italiano	50	120	30	20
Bilingüe Inglés-Español	50	118	32	20
Bilingüe Español-Inglés	50	120	30	20

Tabla 5.2: Número de preguntas del CLEF 2005 de cada tipo para cada colección

pero se desconoce su subcategoría. Las subcategorías que contiene son las siguientes:

NAME.ACRONYM Esta categoría se utiliza para preguntas referentes a las siglas de una organización.

NAME.PERSON Se utiliza para preguntas sobre el nombre de alguna persona en concreto.

NAME.TITLE Preguntas referentes a títulos de películas, obras de teatros, libros, etc.

NAME.LOCATION Una categoría general para nombres relacionados con la geografía. Esta categoría también posee subcategorías y son:

NAME.LOCATION.COUNTRY Para nombres de países.

NAME.LOCATION.CITY Para nombres de ciudades.

NAME.LOCATION.GEOGRAPHICAL Para nombres geográficos como montañas, ríos, contienentes, etc.

DEFINITION Otra categoría general para preguntas que requieren la definición de personas y organizaciones. Se divide en estas dos subcategorías:

DEFINITION.ORGANIZATION Para obtener nombres de organizaciones normalmente a partir de sus siglas.

DEFINITION.PERSON Preguntas referentes a cargos o posiciones de personas.

DATE Preguntas que requieren como contestación alguna fecha. Esta categoría se divide en 4 subcategorías mas:

DATE.DAY Para preguntas que requieren una fecha exacta con el día, mes y año.

DATE.MONTH Preguntas referentes a meses del año.

DATE.YEAR Si no se especifica lo contrario, para las preguntas del CLEF que requieren fechas, son válidas las respuestas referentes al año en que ocurrió el hecho y no hace falta la fecha completa.

DATE.WEEKDAY Para preguntas que piden el nombre de un día de la semana.

QUANTITY Cuando se requiere respuestas sobre cantidades se utiliza esta categoría que se compone de estas subcategorías:

QUANTITY.MONEY Para cantidades que requieran monedas de cualquier país.

QUANTITY.DIMENSION Para cantidades de medida de distancia, peso, etc.

QUANTITY.AGE Para preguntas que requieran edades de personas u objetos.

GENERAL Esta categoría se asigna a preguntas las cuales no se han podido clasificar en las categorías anteriores. En tal caso, el módulo de extracción de la pregunta, podrá extraer cualquier tipo de información.

Como se puede observar, hemos dividido las categorías de preguntas en una arquitectura jerárquica. Esto nos permite, asignar una pregunta a una categoría cuando no encontramos a qué subcategoría pertenece y así aplicar las reformulaciones, los filtros y las reglas de extracción de la respuesta de la categoría más general. En la tabla 5.3 se puede observar el número de preguntas de cada tipo.

Esta clasificación ha sido realizada automáticamente utilizando una Máquina de Soporte Vectorial a las que, posteriormente se le ha aplicado un conjunto de reglas basadas en patrones regulares [GBBA+06].

Tipo de respuesta esperado	Número de preguntas
DATE	13
DATE.DAY	4
DATE.MONTH	0
DATE.WEEKDAY	0
DATE.YEAR	2
DEFINITION	0
DEFINITION.ORGANIZATION	25
DEFINITION.PERSON	26
GENERAL	13
NAME	36
NAME.ACRONYM	1
NAME.LOCATION	9
NAME.LOCATION.CITY	2
NAME.LOCATION.COUNTRY	16
NAME.LOCATION.GEOGRAPHICAL	0
NAME.PERSON	27
NAME.TITLE	2
QUANTITY	16
QUANTITY.AGE	2
QUANTITY.DIMENSION	3
QUANTITY.MONEY	3

Tabla 5.3: Número de preguntas de cada tipo

Finalmente, se han corregido los errores manualmente con el objetivo de obtener la mejor clasificación para una evaluación precisa de los sistemas de RP. Esto nos permitirá conocer, con gran precisión, qué sistemas funcionan mejor para cada tipo de pregunta. Pese a todos los esfuerzos en conseguir una clasificación lo más exacta posible, siempre queda un margen de error debido a la ambigüedad del lenguaje natural. Así, de las 200 preguntas, tenemos 13 que no pertenece a una categoría especial debido a que la respuesta puede aparecer de cualquier forma, por ello, se han incluido en el grupo de GENERAL. Las preguntas del tipo NAME, NAME.PERSON, DEFINITION.PERSON y DEFINITION.ORGANIZATION son las más frecuentes y, ya a bastante

más distancia, las preguntas del tipo NAME.LOCATION.COUNTRY. Si aglutinamos por la categoría más general, la clase NAME es la más abundante, con un total de 93 preguntas de este tipo, seguida de DEFINITION con 51, QUANTITY con 24, DATE con 19 y GENERAL con 13 preguntas.

Finalmente, hemos usado dos colecciones de respuestas realizadas por dos evaluadores humanos usando diferentes sistemas de RP para obtener un amplio rango de respuestas para cada pregunta. Además, se han usado dos criterios distintos para crear las colecciones de respuestas. El primero es un criterio *estricto* el cual tiene en cuenta la respuesta dada por los evaluadores del CLEF y las respuestas que hemos encontrado las cuales estamos bastante seguros que son respuestas correctas. El segundo conjunto de respuestas fue obtenido empleando un criterio *indulgente*. Por ejemplo, para la pregunta "*¿Qué son las FARC?*", el criterio estricto debería aceptar sólo "*Fuerzas Armadas Revolucionarias de Colombia*" mientas que el criterio indulgente también aceptaría como respuestas "*grupo guerrillero*" o "*grupo rebelde*"[2].

5.3. Ajustes de parámetros

En este apartado se han realizado una serie de experimentos para obtener los mejores ajustes de los distintos modelos de n-gramas y, de esta forma, elegir el mejor modelo para futuras comparaciones. Como se describe en el capítulo 4, los modelos de n-gramas requieren un motor de búsqueda como primera etapa basado en un sistema de RP clásico que utilice las palabras claves de la pregunta. Este motor de búsqueda acotará el número de pasajes en la entrada de los modelos de n-gramas reduciendo considerablemente las comparaciones entre la consulta y los distintos pasajes. Es necesario, por tanto, definir qué motor de búsqueda será el mejor como primera etapa y cuántos pasajes debe entregar éste a los modelos de n-gramas. También, para el modelo de Densidad de Distancias de n-gramas, hay que ajustar el factor de distancia modificando el valor de k.

Para comparar el rendimiento de los tres modelos de n-gramas, tal

[2] Ambos conjuntos de respuestas, junto con el sistema JIRS, pueden ser descargado en http://jirs.dsic.upv.es.

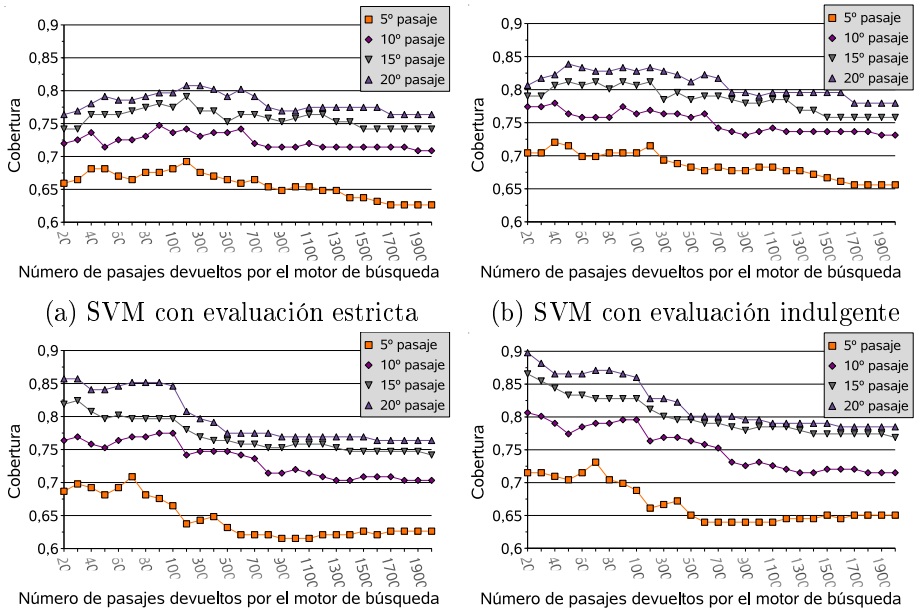
como se describe en el capítulo 4, hemos implementado dichos modelos para el módulo de N-gram Similarity y hemos desarrollado un sistema de RP clásico basado en el tradicional Space Vectorial Model (SVM) para el módulo de Search Engine, así como otro motor de búsqueda que hemos denominado *Relevant Word Density* (*RW-Density*), especialmente adaptado para los modelos de n-gramas. Éste último sólo tiene en cuenta las palabras claves que aparecen en la consulta, sin realizar ningún tipo de normalización del tamaño del pasaje tal y como hace el SVM. Además, hemos generado, para cada uno de los experimentos, una evaluación con el conjunto de respuestas estricto y otro con el conjunto de respuestas indulgente.

5.3.1. Modelo Simple

Los modelos de n-gramas dependen, en gran medida, del motor de búsqueda que se utiliza como primera etapa. Pero no sólo afecta el motor de búsqueda sino también el número de pasajes que éste devuelve. Cuando más pasajes devuelva, más pasajes reordenará los modelos de n-gramas y mayor influencia tendrá éstos. Sin embargo, cuando menos pasajes devuelva el motor de búsqueda, con menos pasajes podrá jugar los modelos de n-gramas y más importancia tendrá el motor. Así, si el motor de búsqueda sólo devuelve 20 pasajes, los modelos de n-gramas sólo podrán reordenar estos 20 pasajes dentro de la lista y devolver dicha lista. En esta situación, los modelos de n-gramas sólo pueden obtener la misma cobertura que obtiene el motor de búsqueda en el vigésimo pasaje. Pero si el motor de búsqueda devuelve unos 2000 pasajes, los modelos de n-gramas reordenarán toda la lista pero sólo devolverán los 20 primeros de la nueva lista. Los siguientes experimentos servirán para evaluar cuál es la mejor cantidad de pasajes que debe devolver los motores de SVM y RW-Density para el modelo Simple.

En la figura 5.1 podemos apreciar cómo varía la cobertura cuando modificamos el motor de búsqueda y el número de pasajes que devuelve. Se ha variado el número de pasajes entre 20 y 2000 puesto que el número de pasajes que devuelve el sistema al final es 20 (independientemente del número de pasajes devueltos por el motor de búsqueda) y no tiene sentido devolver menos. Con más de 2000 pasajes los recursos consumidos son excesivos (recordar que los modelos de n-gramas deben crear conjuntos

de n-gramas para cada pasaje devuelto y compararlos con la consulta). Dentro de los 20 primeros pasajes devueltos se han realizado medidas en el quinto, décimo, décimo quinto y vigésimo pasaje para observar el incremento de cobertura a medida que evaluamos más pasajes.



(a) SVM con evaluación estricta

(b) SVM con evaluación indulgente

(c) RW-Density con evaluación estricta

(d) RW-Density con evaluación indulgente

Figura 5.1: Cobertura en el 5º, 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

Como podemos observar en la figura 5.1, el modelo Simple mejora la cobertura del SVM cuando éste devuelve un número de pasajes que oscila entre 30 y 500. Siendo muy parecida la curva cuando se examina el quinto, décimo y décimo quinto pasaje. Aunque para la evaluación indulgente, el número de pasajes que consigue mejorar la cobertura sube hasta los 700. Las mejoras oscilan entre un 4 % y un 6 % según la posición del pasaje que se observe. Como era de esperar, con la evaluación indulgente se obtienen mejores resultados que con la evaluación estricta puesto que las respuestas de la evaluación indulgente contiene a las respuesta de la evaluación estricta y, además, se añaden otras. De esta forma, es

más probable encontrar pasajes con la respuestas al haber más posibles respuestas.

Tanto en la evaluación estricta como indulgente, a partir de cierto número de pasajes devueltos por el motor de búsqueda, las prestaciones empiezan a disminuir debido a que, el modelo Simple, empieza a encontrar pasajes con n-gramas compuestos por palabras poco relevantes o comunes. Esto es debido a que el modelo Simple prioriza los pasajes con n-gramas largos en contra de pasajes más cortos pero con palabras relevantes. Además, el motor de búsqueda, a medida que aumenta el número de pasajes devueltos por éste, empieza a entregar cada vez más pasajes poco relevantes a la pregunta.

Por otra parte, cuando se utiliza el motor de búsqueda RW-Density como primera etapa, la cobertura no mejora sustancialmente. Es más, si no se escoge con cuidado un número determinado de pasajes devueltos por el motor de búsqueda, es muy probable que los resultados empeoren. Como se aprecian en las gráficas, el RW-Density, por sí solo, mejora en un 10 % al SVM en el vigésimo pasaje, diferencia que se reduce cuando observamos la cobertura en un número de pasaje inferior. En la evaluación indulgente, con el motor RW-Density, los resultados para los modelos de n-gramas no son muy buenos, puesto que a medida que lo aplicamos a más y más pasajes, la cobertura desciende. Solamente, cuando se observa en el quinto pasaje, existe una destacable mejoría de un 2 % pero igualmente a partir de los 90 pasajes el sistema empeora notablemente. Esta mejoría se debe a que encuentra estructuras de la pregunta largas y que, además, contengan las palabras relevantes, entonces el modelo Simple las pondrá en las primeras 5 posiciones del ranking. Esto ocurre tanto en la evaluación estricta como indulgente y nos indica que, para muy pocos pasajes, el modelo Simple funciona bastante bien. Esto nos indica también que, de entre la gran masa de pasajes devueltos, sólo existe unos pocos que tengan expresiones parecidas a la pregunta pero que éstos son muy importantes para obtener buenos resultados. Es de suponer que, en colecciones de documentos mayores, con mayor redundancia, el modelo Simple mejoraría.

Otra clara característica que observamos en las gráficas es que, a medida evaluamos más pasajes, mayor es la probabilidad de encontrar alguno con la respuesta. Así, y por poner un ejemplo, la evaluación indulgente con el motor RW-Density, la cobertura varía desde un 71 % en el

quinto pasaje a casi 90 % en el vigésimo. Esta característica se repite en todos los experimentos realizados independiente del motor de búsqueda, del modelo de n-gramas o de sus parámetros: cuando más pasajes evaluamos más cobertura obtenemos. Esto ha sido así hasta los 200 pasajes por pregunta que hemos analizado, llegando, en este caso, al 96 %.

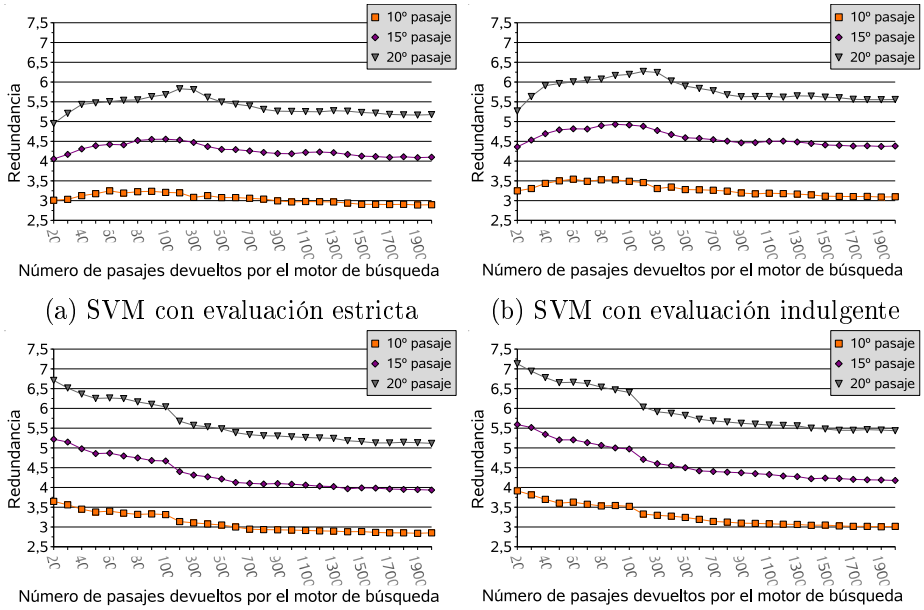
Si observamos ahora la redundancia del sistema (figura 5.2), se aprecia que el modelo Simple mejora la redundancia en todos los casos al SVM, incluso en aquellos en que la cobertura era menor. Esto indica que, el modelo Simple con el motor SVM, como media, encuentra casi 6 pasajes con la respuesta correcta entre los 20 primeros para la evaluación estricta y 6.4 para la evaluación indulgente. Para este cálculo también se ha tenido en cuenta las preguntas para las que JIRS no ha obtenido ningún pasaje con la respuesta. Sólo han sido rechazadas las preguntas que no existe ninguna respuesta en la colección de documentos. La redundancia se ha medido en el décimo, décimo quinto y vigésimo pasaje, y se puede observar que la diferencia entre JIRS y el SVM crece a medida que aumentamos el número de pasajes examinados al igual que ocurría con la cobertura.

Para el modelo Simple, y con el motor SVM, la cobertura puede variar entre 4,9 y 5.81 para la evaluación estricta y 5.21 y 6.26 para la evaluación indulgente. Esto nos indica que el modelo Simple es capaz de encontrar una respuesta más por pregunta que el modelo SVM. Los picos de redundancia, al igual que ocurría con la cobertura, se sitúa alrededor de 200 pasajes.

Sin embargo, cuando utilizamos el motor de búsqueda RW-Density, el modelo de n-gramas empeora la redundancia en relación directa con el número de pasajes devueltos por RW-Density. Según estas gráficas, la presencia de las palabras claves (o pivotes) de la pregunta en los pasajes es muy importante para encontrar la respuesta. Hay que recordar que el motor RW-Density da más valor a aquellos pasajes que contienen mayor número de palabras pivotes y de mayor peso. El modelo Simple, que únicamente tiene en cuenta la longitud de los n-gramas encontrados, no sabe aprovechar este hecho.

El MRR de los sistemas para la evaluaciones estricta y difusa se puede observar en la figura 5.3.

En esta gráfica se ha evaluado el MRR en el quinto pasaje usando tanto el SVM como el RW-Density como motores de búsqueda. Se puede



(a) SVM con evaluación estricta

(b) SVM con evaluación indulgente

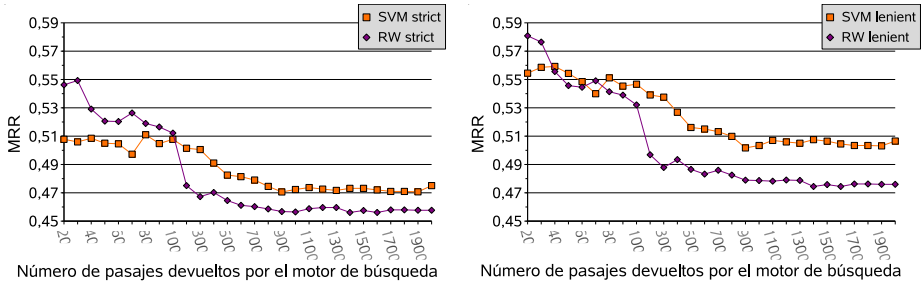
(c) RW-Density con evaluación estricta

(d) RW-Density con evaluación indulgente

Figura 5.2: Redundancia en el 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

apreciar que el MRR baja a medida que añadimos al modelo Simple más pasajes. Esto nos indica que, al añadir ruido, es decir, pasajes pocos relevantes al modelo Simple, éste empeora su cobertura incluso en los primeros 5 pasajes. Es notable la semejanza entre los modelos cuando se evalúan con el conjunto indulgente de respuestas, sobretodo cuando todavía no se ha añadido demasiado ruido al sistema. También hay que destacar la caída desigual de MRR si se usa el motor RW-Density y el motor SVM. La caída de MRR para el RW-Density es más aguda que en el SVM. La caída se acentúa más a partir de los 200 pasajes que cae en picado situándose en un 0.04 o 0.05 (según la evaluación estricta o indulgente respectivamente) por debajo del motor SVM. A partir de ahí, el MRR del modelo con SVM se mantiene constante pero el del RW-Density tiende a bajar.

Si comparamos estos resultados con los de las figuras 5.1 y 5.2 de



(a) Evaluación estricta (b) Evaluación indulgente

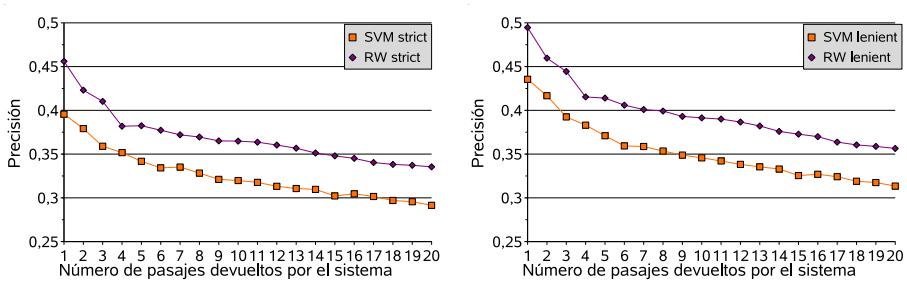
Figura 5.3: MRR en el 5º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

cobertura y precisión respectivamente vemos que, por lo menos para el modelo con SVM, el modelo de n-gramas Simple reordena mejor los primeros 5 pasajes si se le da menos pasajes ruidosos. Al añadir más pasajes al modelo Simple se cuelan pasajes entre los 5 primeros con n-gramas largos pero que no son relevantes, o que no contienen alguna palabra clave para encontrar la respuesta.

La precisión del modelo la podemos observar en la gráfica de la figura 5.4. En esta gráfica se ha evaluado la precisión para los 20 primeros pasajes devueltos por el modelo Simple utilizando los primeros 200 pasajes devueltos por el motor SVM y los 20 primeros cuando se ha utilizado el RW-Density. Como podemos observar la precisión desciende a medida que observamos más pasajes. Esto nos indica que el modelo Simple encuentra más respuestas en los primeros pasajes, tanto con el modelo SVM que con el modelo RW-Density y tanto en la evaluación estricta como la indulgente.

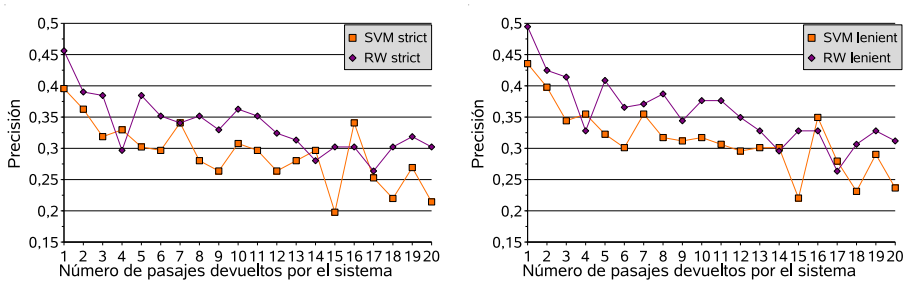
Si, en vez de calcular la precisión acumulativa, observamos la precisión media en los distintos pasajes del ranking, obtenemos la figura 5.5.

En esta figura nos indica en qué número de pasajes se obtienen mejores resultados. Es curioso observar que a la altura del cuarto pasaje se encuentran pocas respuestas en el modelo RW-Density pero, sin embargo, se produce un pico de precisión muy destacado si utilizamos el motor SVM. Pero en general, la precisión, tiende a disminuir a medida que observamos más y más pasajes tal y cómo ocurría con el cálculo de la precisión de la figura 5.4.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.4: Precisión del modelo Simple a medida que aumentamos el número de pasajes examinados



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.5: Precisión por pasaje

En las tablas 5.4 y 5.5 podemos observar la cobertura de la respuesta dividido por tipo de respuesta esperado.

Así, vemos que los peores resultados son para las preguntas del tipo QUANTITY, sobretodo las relacionadas con monedas. Esto es así tanto si se utiliza el motor SVM como el RW-Density, aunque para éste último los resultados son ligeramente mejores. El efecto sobre la cobertura total del sistema no es tan grave puesto que los tipos de QUANTITY.MONEY, QUANTITY.DIMENSION y QUANTITY.AGE son muy poco numerosos. De todas formas sirve para apreciar que el modelo Simple suelen fallar con este tipo de preguntas. En las preguntas del tipo fecha tampoco destaca aunque para el tipo DATE.DAY puede alcanzar el 75 % con la evaluación indulgente. Sin embargo, las preguntas relacionadas con nom-

T. de respuesta esperado	Número de pasajes evaluados								NP
	5	10	15	20	5	10	15	20	
DATE	0,40	0,40	0,40	0,40	0,40	0,40	0,40	0,50	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
DEF.ORG	0,64	0,72	0,84	0,84	0,68	0,72	0,84	0,84	25
DEF.PERSON	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	26
GENERAL	0,33	0,44	0,56	0,56	0,60	0,70	0,70	0,70	13
NAME	0,73	0,83	0,87	0,93	0,74	0,81	0,84	0,90	36
NAME.ACRON	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1
NAME.LOCAT	0,88	0,88	0,88	1,00	0,88	0,88	0,88	1,00	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	1,00	1,00	1,00	1,00	0,94	1,00	1,00	1,00	16
NAME.PERS	0,74	0,78	0,85	0,85	0,74	0,78	0,85	0,85	27
NAME.TIT	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	2
QUANT	0,38	0,38	0,54	0,54	0,36	0,50	0,64	0,64	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,33	0,67	0,67	0,67	0,33	0,67	0,67	0,67	3
QUANT.MON	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3
	Estricta				Indulgente				

Tabla 5.4: Cobertura por tipo de respuesta esperado para el modelo Simple con SVM

bres de ciudades encuentra el 100 % de las respuestas en los primeros 5 pasajes independientemente del motor escogido y también se alcanza el 100 % con el nombre de acrónimos y países. Si utilizamos el motor RW-Density alcanzamos el 100 % con las preguntas del tipo NAME.PERSON y NAME.TITLE (éste último únicamente con la evaluación indulgente) y con el motor SVM las preguntas del tipo NAME.LOCATION. Con las preguntas del tipo DEFINITION.PERSON se alcanzan mejores resultados con el motor SVM que con el RW-Density, obteniendo el 100 % de las respuestas en los primeros 5 pasajes. En las preguntas del tipo GENERAL los resultados son muy similares utilizando el SVM y el RW-Density como motores de búsqueda, aunque el primero mejora al RW-Density en los primeros 5 pasajes de la evaluación estricta, este último es ligeramente

mejor en los últimos pasajes.

T. de respuesta	Número de pasajes evaluados								NP
esperado	5	10	15	20	5	10	15	20	
DATE	0,50	0,50	0,50	0,50	0,60	0,60	0,60	0,60	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,00	0,00	0,00	1,00	0,00	0,00	0,00	1,00	2
DEF.ORG	0,64	0,84	0,88	0,92	0,64	0,88	0,88	0,92	25
DEF.PERSON	0,88	0,88	0,96	0,96	0,92	0,92	1,00	1,00	26
GENERAL	0,33	0,33	0,56	0,56	0,60	0,60	0,80	0,80	13
NAME	0,77	0,84	0,93	0,93	0,74	0,81	0,94	0,94	36
NAME.ACRON	0,00	0,00	0,00	1,00	0,00	0,00	1,00	1,00	1
NAME.LOCAT	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	0,94	1,00	1,00	1,00	0,94	1,00	1,00	1,00	16
NAME.PERS	0,74	0,93	0,96	1,00	0,78	0,93	0,96	1,00	27
NAME.TIT	0,50	0,50	0,50	0,50	0,50	1,00	1,00	1,00	2
QUANT	0,46	0,54	0,54	0,62	0,43	0,64	0,64	0,71	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,67	0,67	1,00	1,00	0,67	0,67	1,00	1,00	3
QUANT.MON	0,00	0,00	0,00	0,33	0,00	0,00	0,00	0,33	3
	Estricta				Indulgente				

Tabla 5.5: Cobertura por tipo de respuesta esperado para el modelo Simple con RW-Density

Una característica que llama la atención es el nombre de los acrónimos, que no encuentra la solución en los primeros 10 pasajes con el motor RW-Density pero sí que la encuentra en los primeros 5 con el motor SVM pese, como se ha visto en las anteriores gráficas, es bastante peor. El resto de tipos se encuentran entre el 80 % y el 100 % de las respuestas encontradas en los primeros 20 pasajes.

Una peculiaridad que al lector le puede llamar la atención es el hecho de que, para algunos tipos de respuesta como QUANTITY o NAME , los valores para la evaluación estricta pueden ser mayores que para la evaluación indulgente. Esto es debido a dos aspectos que influyen en la creación de los conjuntos de respuestas. Una de las razones es que para el

conjunto estricto hemos dado como respuestas válidas a aquellas que los evaluadores CLEF las marcaron como correctas aunque en realidad no lo fueran. Esta es una de las políticas que hemos seguido en la creación de la evaluación estricta. En la evaluación indulgente no seguíamos esta política porque nuestro objetivo era obtener un conjunto de respuestas más flexible pero correcto. La segunda razón es que muchas preguntas que los evaluadores del CLEF la valoraron como respuestas NIL, es decir, que la respuesta no estaba soportada por la colección de documentos, en realidad sí que estaban soportadas y nuestros evaluadores las encontraron. Así, para la evaluación estricta, estas respuestas se marcaron como NIL siguiendo el criterio de los evaluadores del CLEF pero, para la evaluación indulgente, se introdujeron las respuestas correctas. Esto da una pequeña ventaja a la colección de respuestas estricta puesto que las preguntas no soportadas no son evaluadas en los experimentos y estas preguntas suelen ser muy difíciles de responder.

A partir de estas tablas se puede decir que el modelo Simple tiene dificultades a la hora de localizar cantidades y fechas a partir de una pregunta en lenguaje natural. Esto puede ser debido a que las respuestas para este tipo de preguntas se expresen de una forma muy diferente a las preguntas. Además, las preguntas relacionadas con cantidades y fechas suelen ser mucho más complicadas que las relacionadas con las de nombres o definiciones. Por ejemplo, una pregunta típica de definición sería "*¿Qué es BMW?*". Sin embargo, las preguntas de tipo cantidad suelen ser más largas, como por ejemplo: "*¿A cuánto asciende el premio para la ganadora de Wimbledon?*" o "*¿A cuánto dinero ascendió el premio que recibieron Selten, Nash y Harsanyi por el Premio Nobel de Economía?*". A medida que la pregunta se hace más larga, es menos probable encontrar n-gramas en los pasajes que contengan exactamente la expresión de la pregunta. Por ejemplo, ante la pregunta '*¿A cuánto dinero ascendió el premio que recibieron Selten, Nash y Harsanyi por el Premio Nobel de Economía?*" encontramos el siguiente pasaje con la respuesta:

"*Harsanyi , estadounidense de origen húngaro de 74 años , su compatriota Nash , de 66 , y el alemán Selten , de 64 , comparten este año el galardón creado en memoria del magnate sueco Alfredo Nobel y dotado con* ***siete millones de coronas suecas (946.000 dólares)****. Los tres economistas han obtenido el Nobel por su "precursor análisis del equilibrio en la teoría de los juegos no cooperativos", según la justificación que*

leyó el secretario permanente de la Real Academia sueca de Ciencias , Carl Olof Jacobson ."

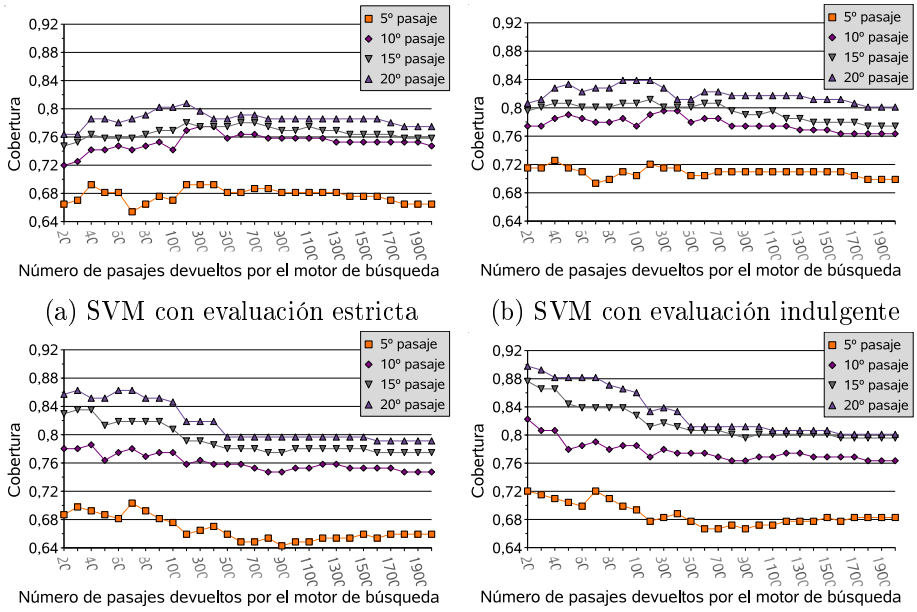
En este pasaje se han subrayado las palabras relevantes de la pregunta y se ha destacado la respuesta en negrita. De esta forma podemos observar que no existe ninguna estructura importante de más de dos palabras de la pregunta y, por lo tanto, nuestra hipótesis sobre el modelo Simple falla para este tipo de preguntas y esta colección. Si esta respuesta fuera más abundante en la colección de documentos, cabría esperar que el modelo Simple pudiera encontrar algún pasaje que contuviera la respuesta en una expresión más similar a la pregunta. Otra opción para subsanar este problema sería realizando una reformulación de la pregunta de forma que tenga más probabilidad de ser encontrada por el modelo Simple.

5.3.2. Modelo Term Weight

Enseguida nos dimos cuenta que el modelo Simple devolvía pasajes con n-gramas largos pero con términos poco importantes en las primeras posiciones. Incluso predominaban este tipo de n-gramas sobre los pasajes con n-gramas más cortos pero con las palabras claves de la pregunta. Esto era debido a que el modelo Simple se basa en contar el número de n-gramas de la pregunta encontrados, sin tener en cuenta los pesos de los términos de los que se compone. El modelo Term Weight fue diseñado para suplir estas deficiencias. En este modelo cada n-grama del pasaje tiene un peso que depende de la importancia de los términos de los que se compone y de los sub-n-gramas que, a su vez, contiene.

Siguiendo el esquema de experimentos realizados para el modelo Simple, el primer experimento se ha realizado para observar cómo afecta a la cobertura del sistema el motor de búsqueda previo al modelo Term Weight. En la figura 5.6 podemos apreciar cómo varía la cobertura del modelo n-gramas en el quinto, décimo, décimo quinto y vigésimo pasaje cuando modificamos el número de pasajes devueltos por el motor de búsqueda tradicional en la etapa anterior. Se ha variado el número de pasajes entre 20 y 2000.

Los resultados son muy similares al modelo Simple. El modelo Term Weight mejora la cobertura de los pasajes devueltos en la vigésima posición por el SVM en todos los casos en la evaluación estricta y hasta 1600 pasajes en la evaluación indulgente. Y, aunque los máximos son ligera-



(a) SVM con evaluación estricta

(b) SVM con evaluación indulgente

(c) RW-Density con evaluación estricta

(d) RW-Density con evaluación indulgente

Figura 5.6: Cobertura en el 5º, 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

mente menores que en el modelo Simple, el modelo Term Weight mejora el SVM en más amplia gama de pasajes devueltos por éste. Como ocurría en el modelo Simple, el Term Weight también tiene el máximo cuando el SVM devuelve 200 pasajes. A medida que observamos menos pasajes, la diferencia entre la evaluación estricta e indulgente se acentúa. Cuando utilizamos el conjunto de respuestas estricto, la cobertura en el décimo, décimo quinto y vigésimo siempre es mejor a medida que aumentamos el número de pasajes devueltos en la primera etapa. Pero en el quinto puede haber grandes varianzas siendo en algunas pocas ocasiones bastante inferior a cuando el SVM sólo devuelve 20. Esto no indica que el modelo Term Weight es peor que el SVM en el quinto pasaje, puesto que los primeros 20 pasajes devueltos por el SVM también son reordenados por el modelo de n-gramas. Pero si comparamos estos resultados con los obtenidos para el modelo Simple de la figura 5.1, vemos que el modelo

Term Weight es más estable en los primeros 5 pasajes y no se aprecia la tendencia a disminuir a medida que aumentamos el número de pasajes devueltos por el SVM.

También, el comportamiento cuando se utiliza el motor RW-Density, es muy parecido en los modelos Simple y Term Weight. En ambos casos, la cobertura empeora indicando que los modelos de n-gramas no mejoran al modelo RW-Density por sí sólo[3].

Estas semejanzas indica que, fundamentalmente, el funcionamiento del modelo Term Weight es muy parecido al modelo Simple. Esto es debido a que la longitud de los n-gramas y su número sigue siendo más importante que la relevancia de los términos que contiene.

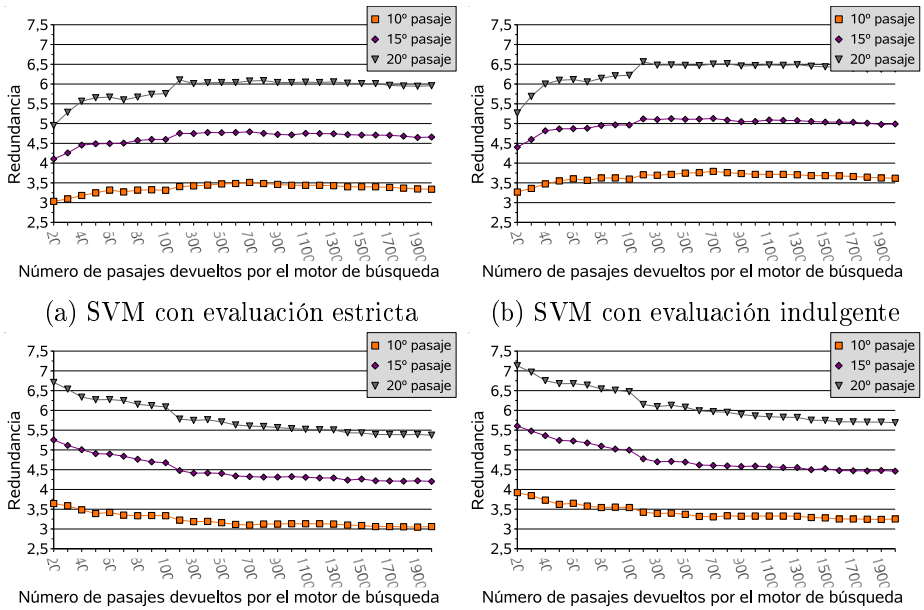
En lo que se refiere a la redundancia, como se aprecia en la figura 5.7, el modelo Term Weight, al igual que el modelo Simple, tiene un máximo cuando el SVM devuelve 200 pasajes, pero al contrario que el anterior modelo, la redundancia no disminuye después y se mantiene constante. Esto ocurre tanto en el décimo, décimo quinto y vigésimo pasaje. También se produce una ligera mejoría en el máximo de un 0.2 de redundancia, tanto en la evaluación estricta como indulgente.

Con el motor RW-Density, la redundancia de la respuesta, también desciende a medida que aumentamos el número de pasajes en la salida del motor de búsqueda. Pero existe una ligera mejoría con respecto al modelo Simple, sobretodo cuando el motor RW-Density devuelve un número elevado de pasajes (entre 300 y 2000).

De todas formas, tanto el modelo Simple como el modelo Term Weight parecen no mejorar, tanto en cobertura como en redundancia, al motor de búsqueda RW-Density basado en palabras claves, aunque mejoran considerablemente la salida del SVM. Ocurre exactamente lo mismo si nos fijamos en el MRR, como se puede observar en la figura 5.8.

Aunque la curva de MRR es muy similar entre el modelo Simple y Term Weight usando los dos motores de búsqueda y a media que aumentamos el número de pasajes devueltos por éstos, el modelo Simple mejora en 0.02 el MRR calculado con los 5 primeros pasajes pero el modelo Term Weight se mantiene más constante a medida que aumentamos el número

[3]Recordemos que la cobertura en el vigésimo pasaje cuando el motor de búsqueda clásico devuelve 20 pasajes coincide con la cobertura del motor búsqueda cuando se aplica sin modelo de n-gramas, no ocurre lo mismo cuando se examina el quinto, décimo y décimo quinto pasaje.



(a) SVM con evaluación estricta

(b) SVM con evaluación indulgente

(c) RW-Density con evaluación estricta

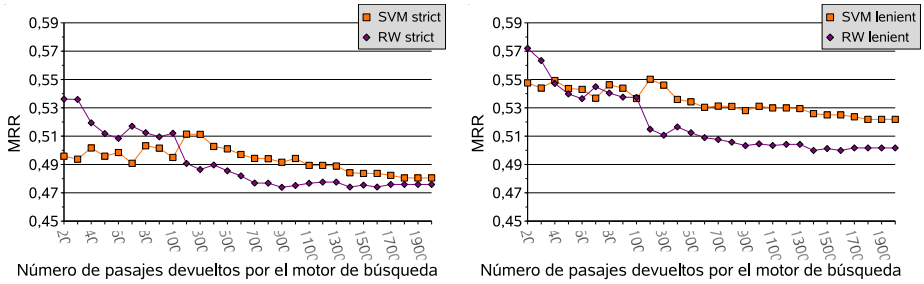
(d) RW-Density con evaluación indulgente

Figura 5.7: Redundancia en el 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

de pasajes del motor de búsqueda. Siendo mejor que el modelo Simple a partir de los 200 pasajes devueltos por el SVM y por el RW-Density. Las curvas de MRR son muy parecidas tanto en la evaluación estricta como indulgente pero, con ésta última, se encuentra entre un 0.05 y 0.02 por encima dependiendo de si se compara el sistema con 20 pasajes devueltos por el SVM o con 2000 pasajes devueltos por el RW-Density.

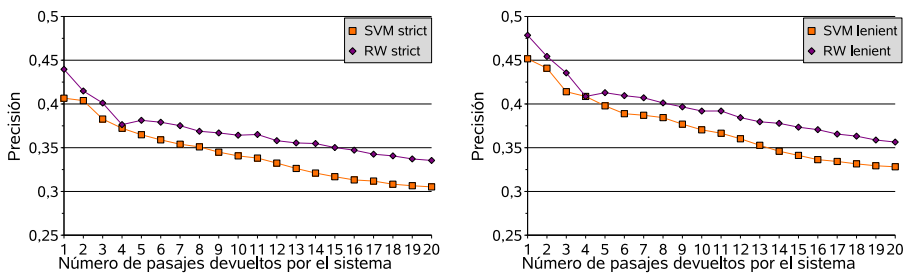
La precisión final del sistema en los primeros 20 pasajes se refleja en la figura 5.9. Para estos experimentos se han utilizado los 200 pasajes devueltos por el SVM y sólo los 20 pasajes devueltos por el RW-Density.

Lo primero que hay que destacar en el cálculo de la precisión es que las curvas, utilizando tanto el SVM como el RW-Density, están más juntas. Siendo destacable que sea el sistema con el SVM el que más aumente con respecto al modelo Simple más que el sistema con RW-Density el que reduzca. Aunque en el primer pasaje es el RW-Density el que desciende



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.8: MRR en el 5º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda



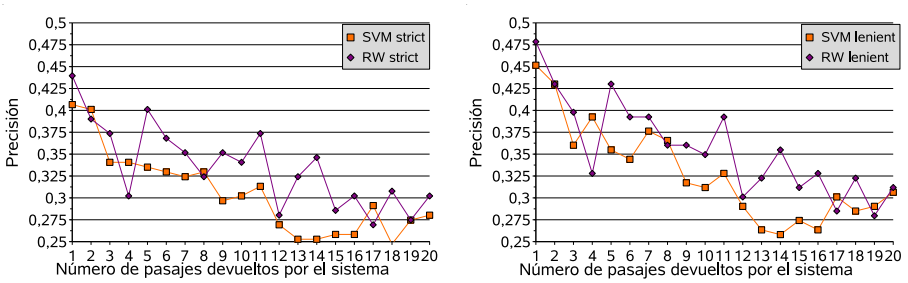
(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.9: Precisión del modelo Term Weight a medida que aumentamos el número de pasajes examinados

más para aproximarse a la precisión con el motor SVM, llegando a reducir un 0.5 % con respecto al mismo experimento con el modelo Simple. Al igual que en el modelo anterior, la precisión desciende, lo cual significa que se encuentra más respuestas correctas en los primeros pasajes. La precisión final indica que, utilizando el RW-Density, se puede encontrar aproximadamente entre un 34 % y un 35 % de pasajes entre los 20 primeros con la respuesta correcta para cada una de las preguntas realizadas al sistema. Este resultado es idéntico cuando se aplica el modelo Simple puesto que estamos utilizando los 20 pasajes del motor RW-Density y, por lo tanto, la precisión en vigésimo pasaje es el mismo.

La precisión por posición del pasaje, como se aprecia en la figura 5.10,

visualiza la precisión no acumulativa en cada posición de los primeros 20 pasajes del ranking. Para este experimento también se han utilizado los mismos parámetros que para la figura 5.9.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.10: Precisión por pasaje

Al igual que en el modelo Simple, la precisión es mayor en los primeros 5 pasajes, aunque aparece el mismo extraño pico descendente en el 4º. Este pico es más agudo en el modelo Term Weight y utilizando el motor RW-Density. En el primer pasaje y en el mejor de los casos se alcanza un 44 % o un 48 % de precisión en el primer pasaje con la evaluación estricta e indulgente respectivamente. Esto indica que, casi la mitad de los pasajes devueltos en la primera posición por los modelos de n-gramas contendrán la respuesta correcta. El mínimo de precisión para cualquier posición de la lista de los primeros 20 pasajes, con el modelo Term Weight, no baja del 25 %, indicando que, en el peor de los casos, uno de cada cuatro pasajes contiene la respuesta correcta. Este mínimo es mayor que en el modelo Simple que puede alcanzar el 20 % en el décimo quinto pasaje con la evaluación estricta.

En general, el modelo Simple es más variable, obteniendo siempre los mejores máximos pero, también, los peores mínimos en las distintas evaluaciones de cobertura y precisión. El Term Weight actúa como un suavizado que mejora los peores resultados del modelo Simple pero empeora ligeramente los mejores. En definitiva el modelo Term Weight es más homogéneo pero sigue teniendo el mismo defecto que el modelo Simple: los n-gramas largos tienen mayor prioridad que los cortos aunque los primeros no contengan palabras relevantes. El peso de un n-grama largo

no depende suficientemente del peso de los términos que contiene. Ésto es debido a cómo se realiza el cálculo del peso de los n-gramas. El peso de cada n-grama es la suma de los términos que contiene más la suma de los pesos de los n-gramas de los que está compuesto. Esto implica una suma recursiva de los mismos términos de un n-grama de tamaño n, de los términos de los n-gramas de tamaño $n-1$ y así hasta la suma de los términos de los unigramas. En la práctica esto implica sumar el peso de un mismo término muchas veces. La veces que se suma un mismo término sólo depende del tamaño del n-grama en el que está contenido. Como veremos, el modelo de Densidad de Distancia de N-gramas suple esta deficiencia. Pero antes de pasar al modelo de Distancia, veamos cómo el modelo Term Weight reacciona para cada tipo de pregunta.

Si nos fijamos en la tabla 5.6, podemos ver la cobertura del Term Weight usando el motor SVM para cada tipo de respuesta esperado.

El Term Weight mejora la cobertura para las preguntas del tipo QUANTITY con la evaluación estricta en los primeros pasajes, aunque no en la evaluación indulgente que sólo lo hace en el 10º y 15º pasaje. Las preguntas del tipo QUANTITY.AGE, al igual que con el modelo Simple, sólo es capaz de encontrar la respuesta a una de las dos preguntas. Con las preguntas del tipo QUANTITY.DIM mejora al modelo Simple en los primeros pasajes, después los resultados son idénticos, Este modelo tampoco es capaz de encontrar la respuesta a ninguna de las 3 preguntas del tipo QUANTITY.MONEY. Ambos modelos son idénticos con las preguntas del tipo DATE y DATE.DAY, aunque con las preguntas del tipo DATE.YEAR sea mejor el Term Weight a partir del 10º pasaje. Con las preguntas del tipo DEFINITION son peores los resultados en los primeros pasajes con el Term Weight si lo comparamos con los resultados del modelo Simple al igual que los resultados en el tipo de pregunta GENERAL. Con el tipo general NAME, el modelo Term Weight es ligeramente mejor, aunque si observamos los subtipos de NAME el modelo Simple es igual o mejor salvo algunas excepciones.

En la tabla 5.7 podemos apreciar los resultados utilizando el motor RW-Density.

Si lo comparamos con el modelo Simple utilizando el motor RW-Density, los resultados son muy parecidos. Aunque hay ligeras mejoras en los tipos DATE.YEAR, GENERAL, NAME, NAME.TITLE (en su evaluación indulgente) y QUANTITY.MONEY y resultados ligeramente

T. de respuesta esperado	Número de pasajes evaluados								NP
	5	10	15	20	5	10	15	20	
DATE	0,40	0,40	0,40	0,40	0,40	0,40	0,50	0,50	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,50	1,00	1,00	1,00	0,50	1,00	1,00	1,00	2
DEF.ORG	0,60	0,80	0,80	0,84	0,64	0,80	0,80	0,84	25
DEF.PERSON	0,96	1,00	1,00	1,00	0,96	1,00	1,00	1,00	26
GENERAL	0,33	0,44	0,44	0,67	0,60	0,70	0,70	0,80	13
NAME	0,83	0,87	0,87	0,90	0,81	0,87	0,87	0,90	36
NAME.ACRON	0,00	0,00	0,00	1,00	0,00	0,00	0,00	1,00	1
NAME.LOCAT	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	0,88	1,00	1,00	1,00	0,88	1,00	1,00	1,00	16
NAME.PERS	0,70	0,81	0,85	0,85	0,74	0,81	0,85	0,85	27
NAME.TIT	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	2
QUANT	0,46	0,46	0,54	0,54	0,43	0,43	0,57	0,64	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,67	0,67	0,67	0,67	0,67	0,67	0,67	0,67	3
QUANT.MON	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	3
	Estricta				Indulgente				

Tabla 5.6: Cobertura por tipo de respuesta esperado para el modelo Term Weight con SVM

peores en DATE y DEFINITION.ORGANIZATION.

Hay mayor diferencia si lo comparamos con la tabla 5.6, en la cual se observa que las preguntas del tipo DATE obtienen 10 % más de cobertura; las preguntas del tipo GENERAL a veces el motor SVM mejora al RW-Density pero otra veces es éste último el mejor; las del tipo NAME mejoran ligeramente, al igual que NAME.ACRONYM en la evaluación indulgente y NAME.COUNTRY. NAME.PERSON mejora entre un 4 % en los primeros pasajes y un 15 % en los últimos; QUANTITY empieza con la misma cobertura pero después mejora en 8 % y QUANTITY.DIMENSION y QUANTITY.MONEY mejoran a partir del 15º pasaje. Sólo las preguntas del tipo DATE.DAY y DEFINITION.PERSON empeoran con respecto al motor SVM.

T. de respuesta esperado	Número de pasajes evaluados								NP
	5	10	15	20	5	10	15	20	
DATE	0,40	0,50	0,50	0,50	0,50	0,60	0,60	0,60	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,00	0,50	0,50	1,00	0,00	0,50	0,50	1,00	2
DEF.ORG	0,60	0,84	0,88	0,92	0,60	0,88	0,88	0,92	25
DEF.PERSON	0,88	0,88	0,96	0,96	0,92	0,92	1,00	1,00	26
GENERAL	0,33	0,44	0,56	0,56	0,60	0,70	0,80	0,80	13
NAME	0,83	0,87	0,94	0,94	0,81	0,84	0,94	0,94	36
NAME.ACRON	0,00	0,00	0,00	1,00	0,00	0,00	1,00	1,00	1
NAME.LOCAT	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	0,94	1,00	1,00	1,00	0,94	1,00	1,00	1,00	16
NAME.PERS	0,74	0,93	0,96	1,00	0,78	0,93	0,96	1,00	27
NAME.TIT	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	2
QUANT	0,46	0,54	0,54	0,62	0,43	0,64	0,64	0,71	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,67	0,67	1,00	1,00	0,67	0,67	1,00	1,00	3
QUANT.MON	0,00	0,00	0,33	0,33	0,00	0,00	0,33	0,33	3
	Estricta				Indulgente				

Tabla 5.7: Cobertura por tipo de respuesta esperado para el modelo Term Weight con RW-Density

5.3.3. Modelo Densidad de Distancias de N-gramas

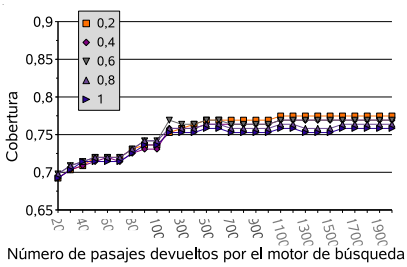
Pese a que el modelo de n-gramas Term Weight fue desarrollado para solucionar los problemas que se observaron con el modelo de n-gramas Simple, éste seguía teniendo los mismos problemas. Principalmente, estos problemas son debido a la necesidad de sumar recursivamente los pesos de los n-gramas. Como los modelos Simple y Term Weight carecen de una medida de distancia entre los distintos n-gramas de la pregunta encontrados en los pasajes, esta suma recursiva era necesaria y difícilmente evitable, puesto que si eliminábamos la suma recursiva sólo teniendo en cuenta el peso de los n-gramas de la pregunta más largos (o con mayor peso) encontrados, daría igual encontrar un n-grama largo, que varios

cortos con la misma longitud para el caso Simple y con los mismos términos para el modelo Term Weight.

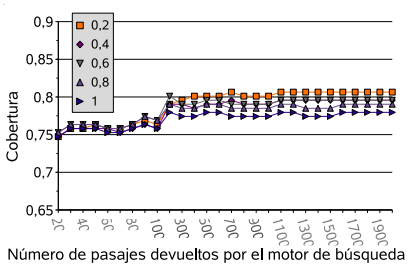
La solución fue desarrollar un modelo de n-gramas que, para calcular el peso de los n-gramas, sólo sumase el peso de los términos una única vez. Para ello, se tuvo que buscar un factor de distancia entre el n-grama de mayor peso, y el resto de n-gramas más cortos. También era importante buscar un algoritmo de similitud entre los n-gramas de la pregunta y del pasaje que tuviera en cuenta que los n-gramas de los pasajes pudieran ser permutaciones de los términos de la pregunta en vez de n-gramas de la pregunta. De esta forma se podría encontrar expresiones en los pasajes que, normalmente, se formulan con los mismos términos que en la pregunta pero en diferente orden. El modelo de Densidad de Distancia de N-gramas (en adelante modelo de Distancia), como se explica en el capítulo 4, realiza una comparación entre el pasaje y la consulta teniendo en cuenta todo lo anterior. Así, para este modelo, no sólo hay que ajustar el motor de búsqueda previo, sino que, además, hay que encontrar el mejor valor de la constante k utilizada para el factor de distancia. Cuando menor sea este valor, menor será la importancia de la distancia en el cálculo de la similitud. Por el contrario, cuando mayor sea el valor de k, mayor importancia adquirirá el factor de distancia de la fórmula 3.10. En las figuras 5.11 y 5.12 se puede observar los resultados de los experimentos para encontrar el valor óptimo de k, el mejor motor de búsqueda previo y el mejor número de pasajes devueltos por éste para el modelo de Distancias.

En este caso, las gráficas han tenido que dividirse en dos: una gráfica realizada con el motor SVM (figura 5.11) y otra realizada con el RW-Density (figura 5.12). Las gráficas se han subdividido en 8 subgráficas en las cuales, la parte izquierda responde a la evaluación estricta y la derecha a la evaluación indulgente. Cada fila corresponde a una medida de cobertura realizada primero en el 5º pasaje y después en el 10º, 15º y 20º del sistema completo para las evaluaciones estricta e indulgente. Al igual que en los modelos Simple y Term Weight, se ha calculado la cobertura para diferente número de pasajes devueltos por el motor de búsqueda (entre 20 y 2000). Por último, y para cada gráfica, se ha evaluado el sistema completo para 5 valores de k que modifica el factor de distancia.

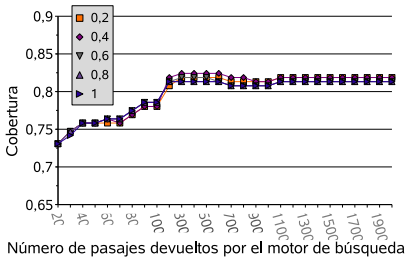
En la figura 5.11 realizada con el SVM, la característica que mejor se observa es que el resultado del modelo de Distancias mejora considerable-



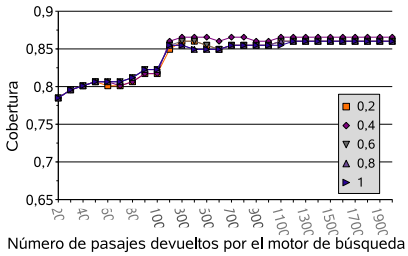
(a) Evaluación estricta en el 5º pasaje



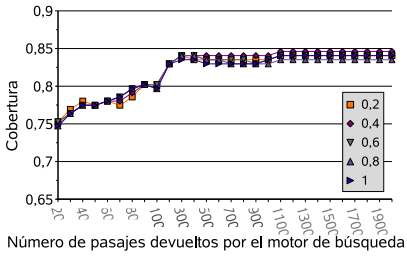
(b) Evaluación indulgente en el 5º pasaje



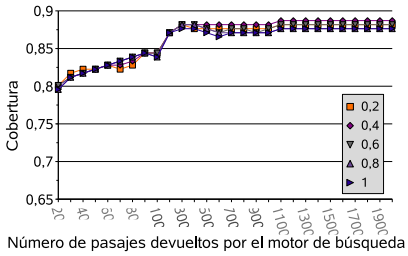
(c) Evaluación estricta en el 10º pasaje



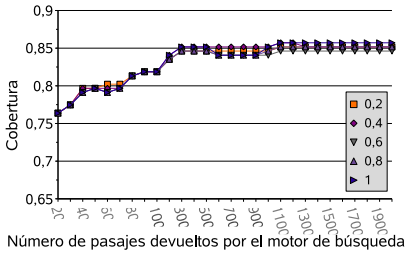
(d) Evaluación indulgente en el 10º pasaje



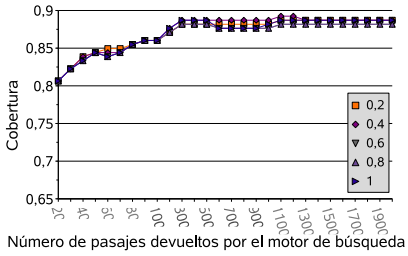
(e) Evaluación estricta en el 15º pasaje



(f) Evaluación indulgente en el 15º pasaje

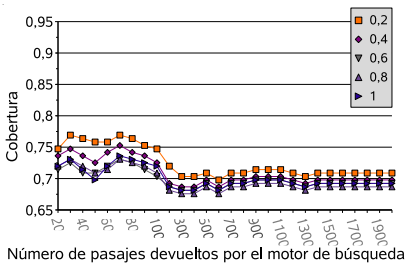


(g) Evaluación estricta en el 20º pasaje

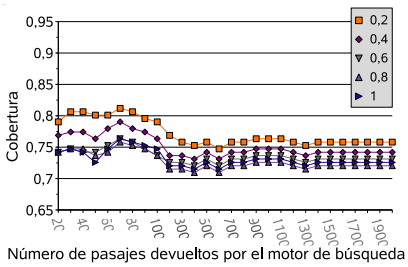


(h) Evaluación indulgente en el 20º pasaje

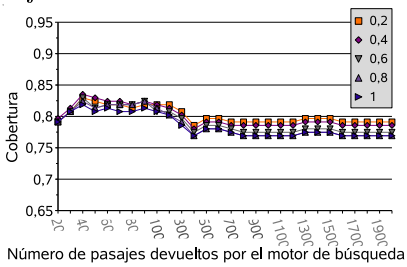
Figura 5.11: Cobertura en el 5º, 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el SVM con factores de distancia de 0.2, 0.4, 0.6, 0.8 y 1



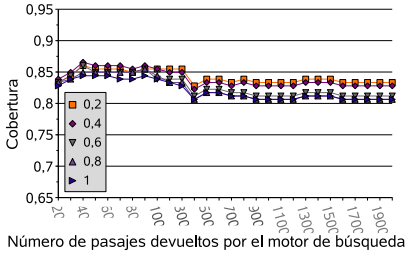
(a) Evaluación estricta en el 5º pasaje



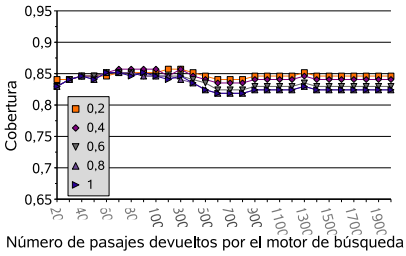
(b) Evaluación indulgente en el 5º pasaje



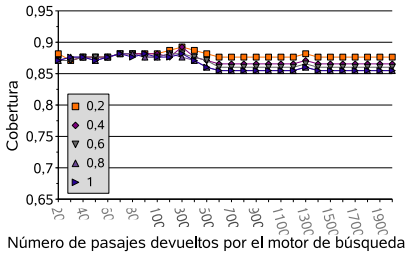
(c) Evaluación estricta en el 10º pasaje



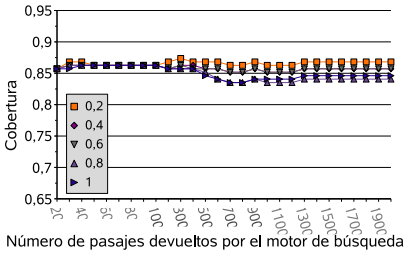
(d) Evaluación indulgente en el 10º pasaje



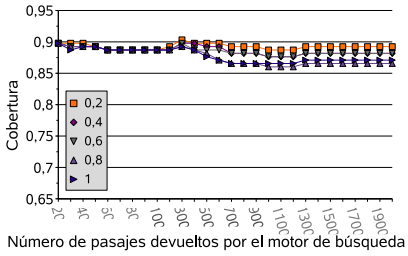
(e) Evaluación estricta en el 15º pasaje



(f) Evaluación indulgente en el 15º pasaje



(g) Evaluación estricta en el 20º pasaje



(h) Evaluación indulgente en el 20º pasaje

Figura 5.12: Cobertura en el 5º, 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el RW-Density con factores de distancia de 0.2, 0.4, 0.6, 0.8 y 1

mente a medida que aumentamos el número de pasajes devueltos por el SVM. Aunque a partir de los 300 pasajes el sistema parece equilibrarse. La diferencia puede alcanzar entre 9 y 10 puntos entre la cobertura devuelta por el SVM en los primeros 20 pasajes y la cobertura del modelo de Distancias cuando reordena 300 o más pasajes del motor de búsqueda para la evaluación estricta e indulgente en el 20º pasaje (subfiguras g y h).

Si nos fijamos en la evaluación estricta (subfiguras a, c, e y g), la cobertura también aumenta a medida que aumentamos el número de pasajes observados, siendo de un máximo de 77 % en el quinto pasaje a máximos de 83 %, 85 % y 86 % en el 10º, 15º y 20º pasaje. Esto indica que la cobertura aumenta mucho si observamos los 10 primeros pasajes pero que reduce su incremento si seguimos observando cada vez más pasajes. Ocurre lo mismo con la evaluación indulgente (subfiguras b, d, f y h), siendo los máximos de 81 %, 87 %, 89 % y 89 % para el 5º, 10º, 15º y 20º pasaje respectivamente.

Con respecto al factor de distancia, vemos que las diferencias para la evaluación estricta e indulgente entre los diferentes valores de k no son muy apreciables. Siendo la diferencia más apreciable de, aproximarse, un 2 % entre el máximo (con k igual a 0.4) y el mínimo (con k igual a 0.6) en el 20º pasaje en la evaluación estricta y de sólo un 1 % en la evaluación indulgente. Si medimos la diferencia de los promedios en el 5º, 10º, 15º y 20º pasaje en vez de únicamene sólo en el último, la diferencia todavía varía menos: sólo en un 1 % en los dos tipos de evaluación.

Para el caso con el motor de búsqueda RW-Density (figura 5.12) las diferencias entre distintos valores de k son más notables, siendo las máximas diferencias cuando se reordenan los 2000 pasajes devueltos por el RW-Density y con la evaluación indulgente. Llegando, en este caso, a alcanzar diferencias de casi 3 %. Las mejoras del modelo de Distancia con nuestro caso base (los primeros 20 pasajes devueltos por el motor de búsqueda) no son tan espectaculares como con el motor SVM mejorando sólo cuando reordena entre 30 y 90 pasajes. A partir de esta cifra los resultados pueden empeoran aunque no en exceso (menos de un 1 % en el 20º pasaje).

Con la evaluación estricta, podemos alcanzar coberturas de 77 %, 84 %, 86 % y 87 % en el 5º, 10º, 15º y 20º pasaje. Unos valores muy parecidos cuando se utiliza el motor SVM en vez del RW-Density, habiendo

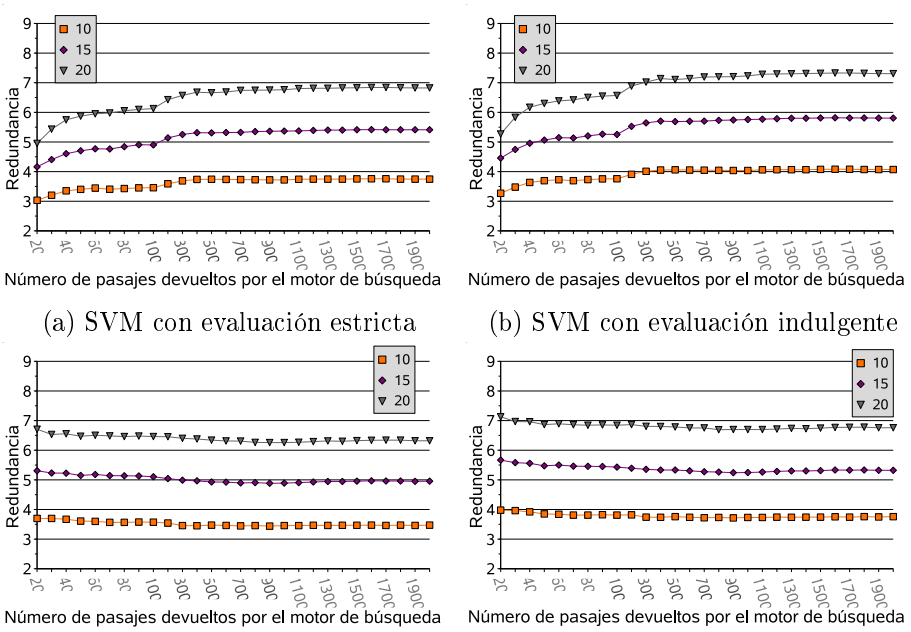
una diferencia de sólo 1 % a partir del 10º pasaje. En la evaluación indulgente obtenemos unas coberturas de 81 %, 87 %, 90 % y 90 % en el 5º, 10º, 15º y 20º pasaje, todavía una diferencia más estrecha con respecto al mismo sistema pero con el motor SVM. Con la evaluación indulgente se obtiene entre un 4 % y un 5 % más de respuestas con respecto a la evaluación estricta.

Para ambos sistemas, aunque las diferencias no son muy apreciables, se obtiene mejores valores de cobertura cuando k oscila entre un 0.2 y un 0.4. Siendo éste último ligeramente superior para la mayoría de los casos. También se ha podido observar que los resultados con el motor RW-Density mejoran a los obtenidos con el SVM, aunque también con una ligera mejoría. En ambos casos, el modelo de Densidad de Distancias de N-gramas mejora con un reordenamiento de pasajes superior a los 20, aunque con el motor RW-Density éste vuelve a empeorar a partir de los 90, cosa que no ocurre con el SVM. También, la semejanza de los mejores resultados nos indica que el funcionamiento del modelo de Distancias trabaja de forma muy parecida usando diferentes motores de búsqueda como primera etapa: si el motor de búsqueda devuelve buenos pasajes, el modelo de Distancias los mantiene en las primeras posiciones y se estabiliza ahí con muy pequeñas diferencias (de menos de un 1 %) aunque se añadan nuevos pasajes pero, sin embargo, si el motor de búsqueda devuelve malos pasajes, el modelo de Distancia los reordena para posicionar en las primeras posiciones pasajes con mayor probabilidad de contener la respuesta.

Para evaluar la redundancia del sistema, y para no complicar en demasía la presente tesis, se calculará los modelos de Distancias con k igual a 0.4 que ha dado muy buenos resultados en las gráficas de cobertura (figuras 5.11 y 5.12).

En la figura 5.13 se puede observar la redundancia del modelo de Densidad de Distancias de N-gramas usando los motores de búsqueda SVM y RW-Density. Si observamos la redundancia a medida que aumentamos el número de pasajes que devuelve el SVM (subfiguras a y b), vemos que la redundancia crece, esto implica que, al aplicar el modelo de Distancias a más pasajes devueltos por el SVM éste es capaz de encontrar más respuestas.

La redundancia del modelo de Distancias usando el motor RW-Density, al contrario que con el motor SVM, empeora a medida que éste devuelve



(a) SVM con evaluación estricta

(b) SVM con evaluación indulgente

(c) RW-Density con evaluación estricta

(d) RW-Density con evaluación indulgente

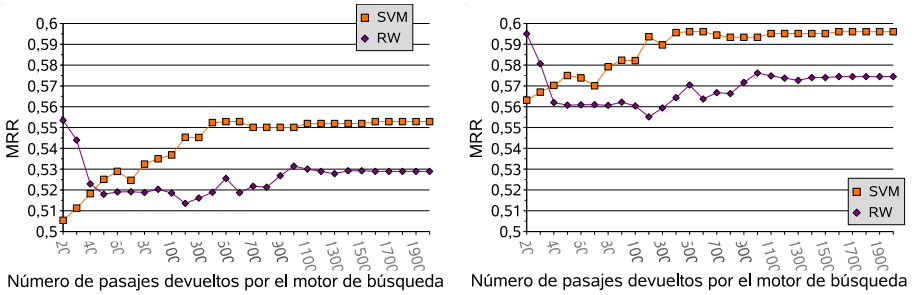
Figura 5.13: Redundancia en el 10º, 15º y 20º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda SVM o RW-Density y con el factor de distancia k de 0.4

más pasajes, aunque muy ligeramente. Incluido en los casos en que la cobertura crecía, como aparece en la figura 5.12. Pero con este motor, la redundancia no alcanza los máximos obtenidos con el motor SVM (6,82 y 7,31 para las evaluaciones estricta e indulgente respectivamente en el 20º pasaje).

En ambos casos, la cobertura crece, obviamente, cuando observamos más pasajes a la salida del sistema. Así, la redundancia del sistema cuando sólo observamos los 5 primeros pasajes es de 2,04, para los 10 primeros es de 3,75 y para los 15 primeros 5,41 en la evaluación estricta y de 2,27, 4,07 y 5,81 respectivamente para la evaluación indulgente. Al igual que ocurría con la cobertura, el mayor incremento de redundancia se produce en los primeros pasajes y va decreciendo a medida que observamos más pasajes. Esto indica que en los primeros pasajes encuentra una propor-

ción mayor de respuestas.

El MRR calculado con los 5 primeros pasajes del modelo de Densidad de Distancias de N-gramas con los motores SVM y RW-Density y con el factor de distancia k de 0.4, se puede observar en la figura 5.14.



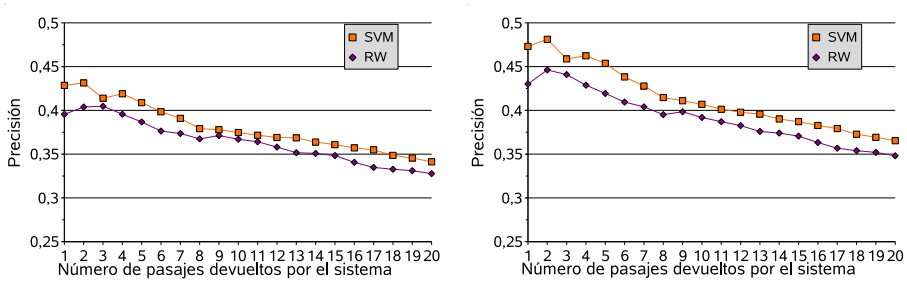
(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.14: MRR en el 5º pasaje a medida que cambiamos el número de pasajes devueltos por el motor de búsqueda

El modelo de Distancia empeora el MRR del motor RW-Density a medida que éste devuelve más pasajes, aunque se estabiliza y mejora a partir de los 600 pasajes. Por el contrario, el MRR mejora si utilizamos el motor SVM a medida que aumentamos el número de pasajes devueltos por el SVM que el modelo de Distancias debe reordenar. El MRR del SVM-Distance llega a igualar o superar al del RW-Distance a partir de ordenar los primeros 400 pasajes devueltos por el motor SVM.

En la figura 5.15 se puede observar la precisión de los modelos. El cálculo de la precisión se ha realizado con 40 pasajes devueltos por el RW-Density para las evaluaciones con el RW-Distance y con 2000 pasajes devueltos por el SVM para el SVM-Distance. En ambos casos, la precisión desciende a medida que observamos más pasajes, indicando ésto que la densidad de respuestas encontradas es mayor en los primeros pasajes. En todas las evaluaciones se muestra un pequeño pico de precisión que se aprecia en el segundo pasaje, indicando que se encuentran un número muy elevado de respuestas correctas si observamos únicamente el segundo pasaje de los 20 primeros que devuelve el modelo de n-gramas.

Lo más destacable es que el modelo de Distancias usando el SVM mejora la precisión cuando se compara con el modelo de Distancias pero

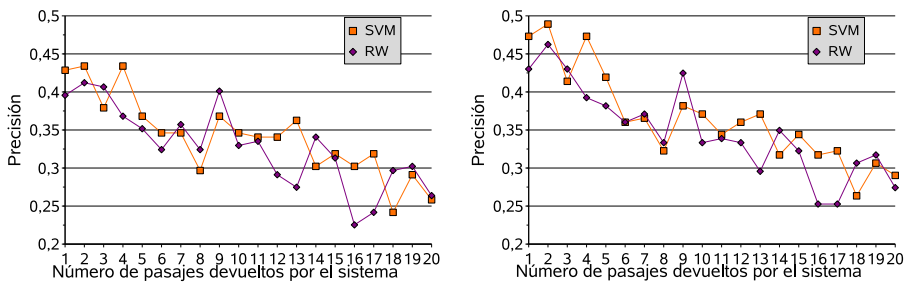


(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.15: Precisión del modelo de Densidad de Distancia de N-gramas a medida que aumentamos el número de pasajes examinados

usando el RW-Density. Este incremento de precisión puede alcanzar un 3 % o un 4 % según realicemos la evaluación estricta o indulgente respectivamente.

Si observamos la precisión por pasaje (figura 5.16), observamos que, para ambos modelos, las precisiones más altas se obtienen en los 4 primeros pasajes, aunque con el motor SVM se produce un pico inferior en el tercer pasaje y con el motor RW-Density en el cuarto.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.16: Precisión por pasaje

En la mayoría de pasajes se obtiene una mejoría en la precisión cuando se utiliza el motor SVM con el modelo de Distancias, pero el motor RW-Density supera al SVM sustancialmente en los pasajes 3º, 10º, 14º y 18º, mientras que el SVM lo hace en el 1º, 2º, 12º, 13º, 16º y 17º.

Para observa en qué tipo de preguntas estos modelos funcionan mejor hay observar las tablas 5.8 y 5.9. Los parámetros utilizados en esta tabla para los distintos modelos son los mismos que para la gráficas de precisión anteriores (figuras 5.15 y 5.16). La primera tabla representa la cobertura para el modelo de Distancias con el motor SVM en los pasajes 5º, 10º, 15º y 20º para las evaluaciones estrictas y difusas.

T. de respuesta	Número de pasajes evaluados								NP
esperado	5	10	15	20	5	10	15	20	
DATE	0,40	0,40	0,40	0,40	0,40	0,60	0,60	0,60	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,50	1,00	1,00	1,00	0,50	1,00	1,00	1,00	2
DEF.ORG	0,76	0,80	0,88	0,88	0,80	0,84	0,88	0,88	25
DEF.PERSON	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	26
GENERAL	0,56	0,67	0,67	0,78	0,80	0,90	0,90	0,90	13
NAME	0,80	0,93	0,93	0,93	0,77	0,94	0,94	0,94	36
NAME.ACRON	0,00	0,00	1,00	1,00	0,00	0,00	1,00	1,00	1
NAME.LOCAT	0,75	0,75	0,88	0,88	0,75	0,75	0,88	0,88	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	16
NAME.PERS	0,93	0,96	1,00	1,00	0,93	0,96	1,00	1,00	27
NAME.TIT	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	2
QUANT	0,54	0,54	0,54	0,54	0,57	0,64	0,64	0,64	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,33	0,67	0,67	0,67	0,33	0,67	0,67	0,67	3
QUANT.MON	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	3
	Estricta				Indulgente				

Tabla 5.8: Cobertura por tipo de respuesta esperado para el modelo de Densidad de Distancias de N-gramas con SVM

Se puede observar que los mejores resultados son para los tipos DEFINITION.PERSON, NAME.CITY y NAME.COUNTRY que encuentran todas las respuestas en los primeros 5 pasajes. Aunque para el tipo NAME.CITY no es destacable al haber sólo dos preguntas de este tipo en la colección de preguntas. Sin embargo, en las preguntas de tipo DEFINITION.PERSON encuentra las respuestas para las 26 preguntas de este

tipo. Una gran cobertura también se consigue con las preguntas del tipo NAME y NAME.PERSON que encuentra el 93 % de las 27 preguntas de esta categoría en los primeros 5 pasajes y el 100 % a partir del 15º pasaje para las NAME.PERSON. Para las preguntas del tipo DATE sólo se encuentra el 40 % con la evaluación estricta y 60 % con la evaluación indulgente. DEFINITION.ORGANIZATION alcanza el 88 % y GENERAL el 78 % o el 90 % según sea la evaluación estricta o indulgente respectivamente. Para las del tipo QUANTITY, al igual que las de DATE, tampoco se consigue buena cobertura, alcanzando sólo el 54 % con la evaluación estricta y 64 % con la indulgente.

En la tabla 5.9 se puede observar la cobertura por tipo de pregunta del modelo de Distancias cuando usa el motor de búsqueda RW-Density.

T. de respuesta	Número de pasajes evaluados								NP
esperado	5	10	15	20	5	10	15	20	
DATE	0,30	0,40	0,40	0,40	0,40	0,50	0,50	0,50	13
DATE.DAY	0,50	0,50	0,50	0,50	0,75	0,75	0,75	0,75	4
DATE.YEAR	0,50	1,00	1,00	1,00	0,50	1,00	1,00	1,00	2
DEF.ORG	0,44	0,60	0,84	0,88	0,48	0,64	0,84	0,88	25
DEF.PERSON	0,88	0,96	1,00	1,00	0,92	1,00	1,00	1,00	26
GENERAL	0,44	0,56	0,67	0,78	0,70	0,80	0,90	0,90	13
NAME	0,87	0,93	0,93	0,93	0,84	0,90	0,90	0,94	36
NAME.ACRON	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	1
NAME.LOCAT	0,75	0,88	0,88	0,88	0,75	0,88	0,88	0,88	9
NAME.CITY	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	2
NAME.COUNT	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	16
NAME.PERS	0,85	1,00	1,00	1,00	0,89	1,00	1,00	1,00	27
NAME.TIT	0,50	0,50	0,50	0,50	1,00	1,00	1,00	1,00	2
QUANT	0,54	0,54	0,69	0,69	0,57	0,64	0,71	0,71	16
QUANT.AGE	0,50	0,50	0,50	0,50	0,50	0,50	0,50	0,50	2
QUANT.DIM	0,33	0,33	0,33	0,33	0,33	0,33	0,33	0,33	3
QUANT.MON	0,33	0,33	0,33	0,67	0,33	0,33	0,33	0,67	3
	Estricta				Indulgente				

Tabla 5.9: Cobertura por tipo de respuesta esperado para el modelo de Densidad de Distancias de N-gramas con RW

El mejor resultado lo da el tipo NAME.COUNTRY pues encuentra el 100 % de las respuestas (16 en total) sólo en los primeros 5 pasajes. Los resultados para los tipos NAME.PERSON y DEFINITION.PERSON (con 27 y 26 preguntas respectivamente) también son muy buenos, alcanzando el 100 % de las respuestas encontradas en el 10º y 15º pasaje. La peor cobertura se obtiene con las preguntas del tipo DATE y QUANTITY pues sólo se alcanzan entre un 40 % y 50 % para las evaluaciones estricta e indulgente del tipo de respuesta esperado DATE y entre 69 % y 71 % para las evaluaciones estricta e indulgente para las del tipo QUANTITY. Las preguntas del tipo DEFINITION.ORGANIZATION también alcanzan buenos resultados llegando a un 88 % tanto en la evaluación estricta como indulgente. Las preguntas del tipo GENERAL varían considerablemente dependiendo del tipo de evaluación, un 78 % para la evaluación estricta y un 90 % para la evaluación indulgente.

El modelo de Densidad de Distancias de N-gramas (independientemente del motor de búsqueda usado), obtiene muy buenos resultados, tanto en cobertura, redundancia, MRR y precisión y para una amplio abanico de tipo de preguntas. Pese a todo, este modelo sigue errando bastante para los tipos de preguntas DATE y QUANTITY debido, seguramente, a la mayor complejidad de las preguntas de estos tipos o a una mayor diferencia entre la pregunta realizada por el usuario y el trozo de texto donde se encuentra la respuesta. Estos modelos fomentan aquellos pasajes que contienen una gran semejanza entre la estructura de la pregunta y la estructura de la frase donde se encuentra la respuesta, aunque las palabras estén en un orden distinto.

5.3.4. Comparación de los modelos de n-gramas

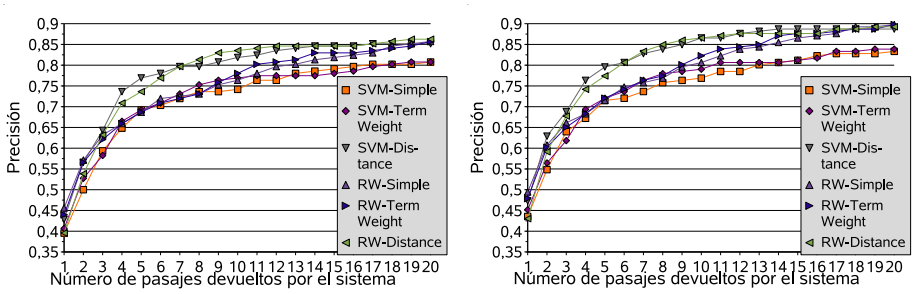
En este apartado vamos a comparar el comportamiento de los distintos sistemas de RP basados en n-gramas vistos en los apartados anteriores. Así, evaluaremos la cobertura, redundancia, MRR y precisión de estos sistemas usando los mejores parámetros de cada uno y con los motores de búsqueda SVM y RW-Density. Además, observaremos la cobertura de cada sistema para cada tipo de pregunta y así determinar para cada tipo de pregunta qué modelo de n-gramas y con qué motor de búsqueda se obtiene los mejores resultados. Esto nos permitirá conocer qué sistemas se deben usar según el tipo de pregunta realizada por el usuario.

Los parámetros utilizados en estos experimentos para cada sistema se pueden consultar en la tabla 5.10.

Modelo de n-gramas	MB	Pasajes MB	k
Simple	SVM	200	—
	RW-Density	20	—
Term Weight	SVM	200	—
	RW-Density	20	—
Distancias	SVM	2000	0.4
	RW-Density	40	0.4

Tabla 5.10: Parámetros utilizados en la comparación de los sistemas: modelo de n gramas usado, motor de búsqueda (MB) previo, pasajes devueltos por el MB y la constante k usada en los modelos de Distancia.

En la figura 5.17 se puede observar las coberturas de los distintos sistemas.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.17: Comparación de la cobertura para los modelos de n-gramas con los motores SVM y RW-Denstiy.

La mejor cobertura la obtienen los sistemas SVM-Distance y RW-Distance aunque RW-Simple y RW-Term Weight son ligeramente mejores en los 3 primeros pasajes. Pese a que el RW-Simple y el RW-Term Weight tienen una cobertura excelente en el primero y último pasaje, hay una zona entre el 4º y el 14º pasaje que la cobertura es muy inferior a la

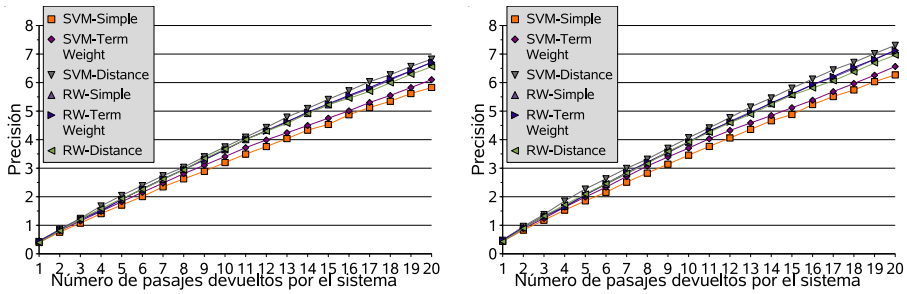
de los sistemas SVM-Distance y RW-Distance, llegando a alcanzar una diferencia de 8 ó 9 puntos dependiendo si la evaluación es estricta o indulgente respectivamente. La cobertura puede alcanzar el 90 % en el 20º pasaje con los sistemas RW-Simple y RW-Term Weight, seguidos muy de cerca (con un 89 %) por los dos modelos de Distancias en la evaluación indulgente. Como era de esperar, en la evaluación estricta la cobertura es inferior en todos los sistemas, llegando a un 86 % en el 20º pasaje para los sistemas RW-Simple, RW-Term Weight y RW-Distance y de 85 % para el sistema SVM-Distance. Los modelos SVM-Simple y SVM-Term Weight son los que obtienen la cobertura más baja en los 20 primeros pasajes, alcanzando únicamente un 81 % en la evaluación estricta y entre 83 % y 84 % en la evaluación indulgente para los modelos Simple y Term Weight respectivamente.

Si observamos la gráfica de redundancia de la figura 5.18, se aprecia que el modelo que obtiene mayor redundancia, tanto en la evaluación estricta como indulgente, es el de Distancias compuesto por el motor de búsqueda SVM, seguido de los modelos Simple y Term Weight pero con el motor RW-Density. Estos últimos tienen un comportamiento tan similar en la redundancia que apenas se aprecia en la gráfica. En la cuarta posición se situaría el RW-Distance y ya, con más diferencia, los modelos SVM-Term Weight y SVM-Simple. Con el mejor modelo (SVM-Distance) se puede alcanzar una redundancia de 6,82 en la evaluación estricta y 7,31 en la evaluación indulgente mientras que, con el peor modelo, en el 20º pasaje obtenemos una redundancia de 5,83 y de 6,27 para las evaluaciones estricta e indulgente respectivamente.

En la tabla 5.11 se muestran los valores de MRR en el 5º pasaje para los distintos sistemas y para las evaluaciones estricta e indulgente.

El sistema con mayor MRR es el sistema SVM-Distance con un 0,55 y un 0,60 de MRR en las evaluaciones estricta e indulgente. El SVM-Simple le sigue muy de cerca igualando el MRR en la evaluación estricta pero alcanzando sólo un 0,58 en la indulgente. Los siguientes sistemas son el RW-Term Weight con un MRR de 0,01 inferior al SVM-Simple y el RW-Distance con un MRR de 0,52 y 0,56. El sistema con peor MRR, al igual que con la cobertura y redundancia, es el SVM-Simple que sólo alcanza un MRR de 0,50 y 0,54 para las evaluaciones estricta e indulgente respectivamente.

La comparativa de precisión se muestra en la figura 5.19.



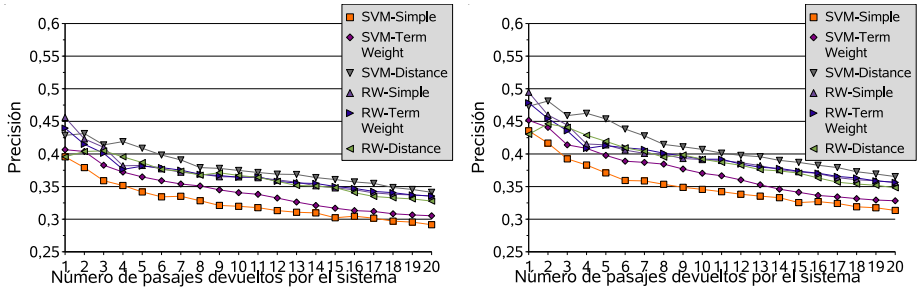
(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.18: Comparación de la redundancia para los modelos de n-gramas con los motores SVM y RW-Denstiy

	Evaluación	
Modelo	Estricta	Indulgente
SVM-Simple	0,50	0,54
SVM-Term Weight	0,51	0,55
SVM-Distance	0,55	0,60
RW-Simple	0,55	0,58
RW-Term Weight	0,54	0,57
RW-Distance	0,52	0,56

Tabla 5.11: Comparativa del MRR en el 5º pasaje para los distintos sistemas basados en n-gramas

En la curva de precisión se observa que el SMV-Distance es el mejor en todos los puntos de la gráfica excepto en el primero que lo superan el RW-Simple y el RW-Term Weight. El sistema RW-Distance empieza bastante mal, igualando o empeorando al peor sistema (SVM-Simple), pero enseguida mejora equiparándose a los sistemas RW-Simple y RW-Term Weight. Estos tres sistemas mantienen su semejanza a lo largo del resto de la curva hasta el 20º pasaje. Aunque la cobertura del SVM-Term Weight era idéntica a la del SVM-Simple, aquí se puede apreciar que su curva de precisión es bastante mejor, aunque no llega a la precisión del resto de los sistemas. El modelo Simple con el motor SVM es también el peor en precisión en todos los puntos de la curva excepto en el primero,

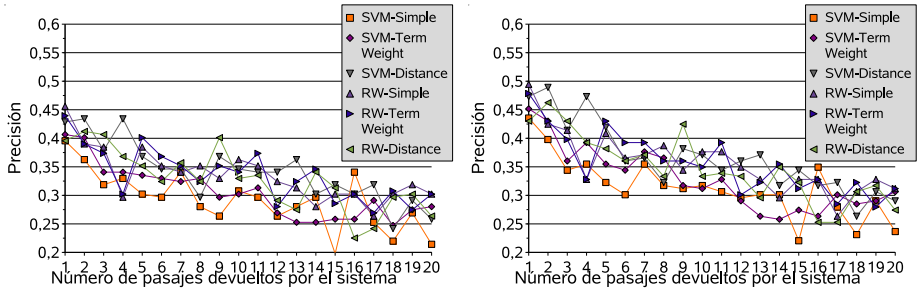


(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.19: Comparación de la precisión para los modelos de n-gramas con los motores SVM y RW-Denstiy

que se iguala al RW-Distance.

La precisión en cada posición de la lista de pasajes devueltos por los sistemas se puede observar en la figura 5.20.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.20: Precisión por pasaje para los modelos de n-gramas con los motores SVM y RW-Denstiy

El SVM-Distance es el que, como media, mejor resultados se obtiene, consiguiendo la mejor precisión en los pasajes 2º, 4º, 12º, 13º, 15º y 17º tanto en la evaluación estricta como indulgente y sin alcanzar un mínimo en ninguno de los pasajes. El sistema RW-Distance obtiene la mejor posición únicamente en el 3º, 7º y 9º pasaje con la evaluación estricta pero pierde la mejor posición en 9º pasaje con la evaluación indulgente. Este sistema también obtiene 3 peores resultados con el 1º,

16º y 17º pasaje en la evaluación estricta y sólo los dos primeros con la evaluación indulgente. El sistema con un número mayor de pasajes con la mejor cobertura es el RW-Term weight pero sólo en la evaluación indulgente (el 5º, 6º, 7º, 11º, 14º, 18º y 20º pasaje) pero pierde el 4º y el 7º con el conjunto de respuestas estricto. Aunque este sistema no alcanza ningún valor mínimo con la evaluación estricta, sí que alcanza dos (el 4º y el 19º) con la evaluación indulgente. El istema SVM-Term Weight no alcanza ningún valor máximo en ningún pasaje pero sí que alcanza mínimos en los pasajes 7º, 10º, 13 y 14º con la evaluación estricta y el 10º, 12º, 13 y 14º con la evaluación indulgente. El modelo Simple con el motor RW-Density (RW-Simple) alcanza máximos en los pasajes 1º, 5º, 8º, 10º, 19º y 20º con la evaluación estricta pero pierde el 5º con la evaluación indulgente. Su contrapartida con el el motor SVM funciona mucho peor, sólo alcanzando un máximo de precisión en el pasaje 18º en ambas evaluaciones pero alcanzando el mayor número de mínimos en innumerables pasajes tanto con la evaluación estricta como indulgente.

Si tuvieramos en cuenta el mejor sistema en cada una de las posiciones de la lista de pasajes, podríamos obtener una mejora en la precisión media de 2 % con respecto al sistema SVM-Distance. Esto indica que el sistema SVM-Distance, por si sólo, consigue casi la misma precisión media en los primeros 20 pasajes que usando los mejores resultados de todos los sistemas. En la misma gráfica observamos una tendencia de todos los sistemas a reducir la precisión a medida que aumentamos el número de pasajes consultados, es decir, que tienden a colocar los pasajes con la respuesta en las primeras posiciones siendo menos probables encontrarlos en posiciones más alejadas.

En la tabla 5.12 hemos desglosado la cobertura en el 5º, 10º, 15º y 20º pasaje para cada grupo general de preguntas. Esto nos permitirá conocer qué sistema funciona mejor y con qué tipo de preguntas.

En las preguntas del tipo DATE, los sistemas SVM-Simple, RW-Simple, SVM-Term Weight y SVM-Distance empiezan muy bien con la evaluación estricta, pero sólo el RW-Simple consigue tener la mejor cobertura en la evaluación indulgente en los primeros 5 pasajes. En el 20º pasaje, los mejores sistemas son RW-Simple y RW-Term Weight en la evaluación estricta y RW-Simple, RW-Term Weight y SVM-Distance en la evaluación indulgente. En los pasajes 10º y 15º, las mejores coberturas son para SVM-Term Weight, RW-Term Weight, SVM-Distance y

	Modelo	Passage number							
	5	10	15	20	5	10	15	20	
DATE (19 p)	SVM-Simple	**0,44**	0,44	0,44	0,44	0,50	0,50	0,50	0,50
	RW-Simple	**0,44**	0,44	0,44	**0,56**	**0,56**	0,56	0,56	**0,69**
	SVM-Term Weight	**0,44**	**0,50**	**0,50**	0,50	0,50	0,56	0,63	0,63
	RW-Term Weight	0,38	**0,50**	**0,50**	**0,56**	0,50	0,63	0,63	**0,69**
	SVM-Distance	**0,44**	**0,50**	**0,50**	0,50	0,50	**0,69**	**0,69**	**0,69**
	RW-Distance	0,31	**0,50**	**0,50**	0,50	0,44	0,63	0,63	0,63
DEFINITION (51 p)	SVM-Simple	0,82	0,86	0,92	0,92	0,84	0,86	0,92	0,92
	RW-Simple	0,76	0,86	0,92	0,94	0,78	0,90	**0,94**	**0,96**
	SVM-Term Weight	0,78	0,90	0,90	0,92	0,80	0,90	0,90	0,92
	RW-Term Weight	0,74	0,86	0,92	0,94	0,76	0,90	**0,94**	**0,96**
	SVM-Distance	**0,88**	0,90	**0,94**	0,94	**0,90**	**0,92**	**0,94**	0,94
	RW-Distance	0,82	**0,92**	**0,94**	**0,96**	0,84	**0,92**	**0,94**	**0,96**
GENERAL (13 p)	SVM-Simple	0,33	0,44	0,56	0,56	0,60	0,70	0,70	0,70
	RW-Simple	0,33	0,33	0,56	0,56	0,60	0,60	0,80	0,80
	SVM-Term Weight	0,33	0,44	0,44	0,67	0,60	0,70	0,70	0,80
	RW-Term Weight	0,33	0,44	0,56	0,56	0,60	0,70	0,80	0,80
	SVM-Distance	**0,56**	**0,67**	**0,67**	**0,78**	**0,80**	**0,90**	**0,90**	**0,90**
	RW-Distance	0,44	0,56	**0,67**	**0,78**	0,70	0,80	**0,90**	**0,90**
NAME (93 p)	SVM-Simple	0,79	0,85	0,88	0,92	0,79	0,85	0,89	0,92
	RW-Simple	0,79	0,88	0,93	**0,95**	0,79	0,89	0,95	**0,97**
	SVM-Term Weight	0,79	0,86	0,87	0,90	0,80	0,87	0,89	0,91
	RW-Term Weight	0,81	0,90	0,93	**0,95**	0,83	0,90	0,95	**0,97**
	SVM-Distance	**0,86**	0,92	**0,95**	**0,95**	0,86	0,93	**0,97**	**0,97**
	RW-Distance	**0,86**	**0,94**	0,94	**0,95**	**0,87**	**0,94**	0,94	**0,97**
QUANTITY (24 p)	SVM-Simple	0,33	0,38	0,48	0,48	0,32	0,45	0,55	0,55
	RW-Simple	0,43	0,48	0,52	**0,62**	0,41	0,55	0,59	**0,68**
	SVM-Term Weight	0,43	0,43	0,48	0,48	0,41	0,41	0,50	0,55
	RW-Term Weight	0,43	0,48	**0,57**	**0,62**	0,41	0,55	**0,64**	**0,68**
	SVM-Distance	**0,48**	0,52	0,52	0,52	**0,50**	0,59	0,59	0,59
	RW-Distance	**0,48**	**0,57**	**0,57**	0,57	**0,50**	**0,64**	**0,64**	0,64
		Estricta				Indulgente			

Tabla 5.12: Cobertura por tipo de respuesta esperado para los sistemas de n-gramas con los motores SVM y RW-Density

RW-Distance para la evaluación estricta y el SVM-Distance para la evaluación indulgente. En general, para el tipo de preguntas DATE, los sistemas que funcionan mejor son el RW-Term Weight y el SVM-Distance aunque este último no consiga ser el mejor de cobertura en el 20º pasaje de la evaluación estricta. Pese a todo, los resultados con las preguntas del tipo DATE no son muy satisfactorias con ningún sistema alcanzando sólo porcentajes de 56 % para la evaluación estricta y 69 % para la evaluación indulgente.

En las preguntas tipo DATE hay mucha varianza entre los resultados en la evaluación estricta y difusa, cosa que no ocurre con los tipos de preguntas DEFINITION. En este tipo de datos el RW-Distance y el SVM-Distance son los claros vencedores en ambas evaluaciones aunque los sistemas RW-Simple y RW-Term Weight consiguen muy buena cobertura en el 20º pasaje. En las preguntas del tipo DEFINITION, se alcanzan coberturas elevadas tanto en la evaluación estricta (94 %) como en la evaluación indulgente (96 %). Además, también se consiguen coberturas muy buenas con los primeros pasajes, obteniendo con el sistema SVM-Distance un 88 % en los primeros 5º pasajes.

Las preguntas del tipo GENERAL tienen variaciones de cobertura muy elevadas dependiendo de si se realiza con la evaluación estricta o indulgente, pero los sistemas que mejor funcionan lo hacen para ambas evaluaciones. Así, para este tipo de preguntas, los mejores sistemas vuelven a ser el SVM-Distance y el RW-Distance, siendo el primero mejor o igual que el RW-Distance en todos los casos. Con este tipo de preguntas, que por otra parte es el que menos preguntas contiene, se obtienen precisiones de entre 78 % y 90 % dependiendo de si la evaluación es estricta o difusa respectivamente. Estos resultados hay que tomarlos con mucha cautela puesto que el tipo de preguntas GENERAL no corresponde a un tipo en concreto, más bien son aquel tipo de preguntas que no encajan en ninguno de los otros tipos y, por lo tanto, pueden ser muy variadas y, dado el bajo número de ese tipo de preguntas, los resultados puede que varíen mucho si lo aplicamos a otro conjunto de preguntas.

El tipo de pregunta con el que se obtienen los mejores resultados es el tipo NAME. Con este tipo se pueden alcanzar precisiones de 95 % con la evaluación difusa y de 97 % con la evaluación indulgente. Los sistemas RW-Simple, RW-Term Weight, SVM-Distance y RW-Distance son los que alcanzan estos máximos tanto en la evaluación estricta como indulgente.

Pero son los sistemas SVM-Distance y el RW-Distance los que obtienen los mejores resultados a lo largo de todos los pasajes.

Otro tipo de preguntas difícil es el tipo QUANTITY. De las 24 preguntas sólo se encuentran respuestas en el 62 % y el 68 % de las preguntas con las evaluaciones estrictas y difusas en los 20 primeros pasajes. En este tipo de preguntas, el SVM-Distance no destaca, haciéndolo el RW-Simple y el RW-Term Weight, aunque el RW-Distance lo hace en los 15 primeros pasajes.

5.3.5. Tamaño del pasaje

Aunque no se ha mencionado con anterioridad, existe un parámetro que afecta a todos los sistemas evaluados: el tamaño del pasaje. Como se ha explicado en el capítulo 4, JIRS construye los pasajes basándose en el número de frases que contiene, en vez de en el número de palabras o párrafos. [LP01] muestra que este método de división de pasajes mejora el rendimiento de los sistemas de RP con respecto a los sistemas que usan un modelo de ventana clásico.

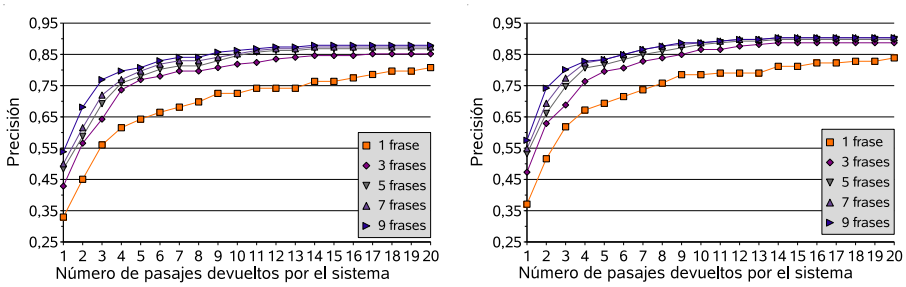
Para poder modificar el tamaño de los pasajes en cada experimento, JIRS permite construir los pasajes en tiempo real, es decir, que crea los pasajes en el momento en que el usuario realiza una consulta en vez de hacerlo cuando se indexa la colección de documentos. Para poder realizar esto de forma eficiente, JIRS indexa la colección de documentos por frases en vez de por pasajes. Así, cuando lanzamos una búsqueda por alguno de los métodos clásicos basados en palabras claves, obtenemos la frase que contiene dichos términos, después le añadimos m frases anteriores y posteriores. Por lo tanto, el tamaño de un pasaje medido en frases es impar puesto que añadiemos el mismo número de frases anteriores y posteriores. Esto tiene excepciones cuando la frase se encuentra al principio del documento, al final, o el documento es demasiado pequeño, en tal caso los pasajes son más pequeños.

Es evidente que, al aumentar el tamaño de los pasajes, la cobertura mejore, puesto que en un pasaje largo es más probable *capturar* la respuesta, es decir, que este pasaje contenga la respuesta. Pero también es evidente que, para el módulo de extracción de la respuesta de un sistema de BR completo, en un pasaje largo es más complicado extraer la respuesta pues contiene mucha más información que pasajes más cortos.

Así, debemos encontrar pasajes que, no siendo demasiado grandes, tengan la suficiente cobertura para poder obtener buenos resultados en la BR.

En los anteriores experimentos hemos realizado todos los experimentos con pasajes de tamaño de 3 frases. Pero en este apartado vamos a variar este parámetro para observar cómo varía la cobertura, la redundancia, el MRR y la precisión a medida que aumentamos o disminuimos el tamaño del pasaje. Para evaluar este parámetro se ha utilizado únicamente el modelo Distance con el motor de búsqueda SVM y con los mejores parámetros obtenidos para este sistema en el apartado anterior. El resto de sistemas tienen un comportamiento similar al que aquí se expone.

En la figura 5.21 podemos observar la cobertura con pasajes de 1, 3, 5, 7 y 9 frases.



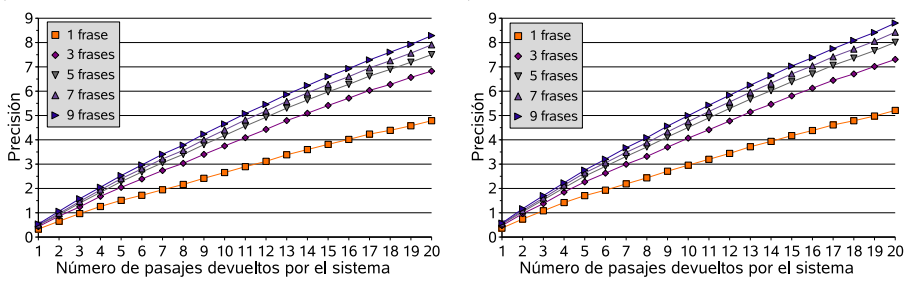
(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.21: Comparación de la cobertura para diferentes tamaños de pasaje

Aunque en líneas generales la cobertura aumenta a medida que aumentamos el tamaño del pasaje, la mayor diferencia se observa entre pasajes de 1 y 3 frases. En este caso la cobertura puede llegar a tener una diferencia de 13 puntos en el 5º pasaje en la evaluación estricta y 11 puntos en el mismo pasaje en la evaluación indulgente. Las diferencias ya no son tan considerables si evaluamos pasajes de 3, 5, 7 y 9 frases, aunque éste último mejora bastante la cobertura en los primeros 4 pasajes, llegando a alcanzar diferencias entre 5 y 6 puntos con respecto al de 7 frases en el 2º pasaje y con la evaluación indulgente y estricta respectivamente. Esto indica que algunas respuestas no se encuentran exactamente

en la misma frase donde aparecen las palabras claves de la pregunta, sino que, con frecuencia, aparecen en la frase anterior o posterior, pero que a medida que nos alejamos de las palabras claves no es tán fácil encontrar nuevas respuestas. Así, con pasajes de tamaño de 5, 7 y 9 frases se puede obtener, en el 20º pasaje, entre 87 % y 88 % de cobertura en la evaluación estricta y 90 % con la evaluación indulgente. Es más, con pasajes de 3 frases la cobertura, en 20º pasaje, sólo se ve reducida un 2 % en la evaluación estricta y un 1 % en las evaluaciones indulgente.

La redundancia también aumenta en ambos casos: a medida que se examinan más pasajes y éstos son de mayor tamaño.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.22: Comparación de la redundancia para diferentes tamaños de pasaje

Como se observa en la figura 5.22, las mayores diferencias de redundancia aparecen cuando comparamos pasajes de 1 y de 3 frases, llegando a encontrar en éste último más de dos respuestas por pregunta en el 20º pasaje tanto en la evaluación estricta como indulgente. Aquí también se aprecia mayor diferencia de redundancia entre los pasajes de tamaño 3 y 5 y los pasajes de tamaño 7 y 9, pero la redundancia entre los tamaños 5 y 7 apenas cambia. Utilizando pasajes con 9 frases, podemos alcanzar aproximadamente entre 8 y 9 respuestas por pregunta dependiendo de si hacemos la evaluación estricta o indulgente. Sin embargo, con pasajes de 1 frase, sólo obtenemos 4,79 respuestas por pregunta para la evaluación estricta y 5,21 para la evaluación indulgente.

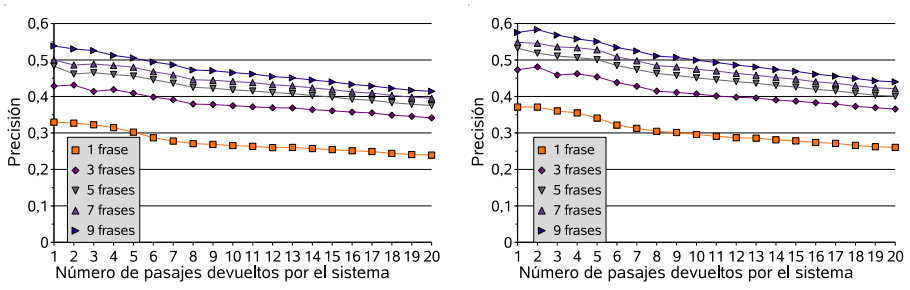
El MRR para diferentes tamaños de pasaje se puede encontrar en la tabla 5.13.

	Evaluación	
Tamaño del pasaje	Estricta	Indulgente
1 frase	0,45	0,50
3 frases	0,55	0,60
5 frases	0,59	0,64
7 frases	0,61	0,66
9 frases	0,65	0,69

Tabla 5.13: Comparativa del MRR en el 5º pasaje para distintos tamaños de pasaje

El comportamiento del MRR es similar al de la cobertura y al de la redundancia. Cuando evaluamos el sistema con pasajes de una frase, el resultado es hasta 10 puntos por debajo de pasajes con 3 frases. Sin embargo, la diferencia es sólo de 4 si comparamos el MRR con pasajes de 3 y 5 frases. Otra vez, la menor diferencia se produce entre pasajes de 5 y 7 frases y aumenta de nuevo cuando comparamos los pasajes de 7 y 9 frases. El MRR, usando el sistema con pasajes de tamaño de 9 frases, alcanza el 0,65 con la evaluación estricta y casi el 0,7 con la evaluación indulgente. Mientras que con una frase, sólo alcanzamos 0,45 y 0,50 respectivamente.

La precisión para cada uno de los distintos tamaños de pasaje se muestran en la figura 5.23.

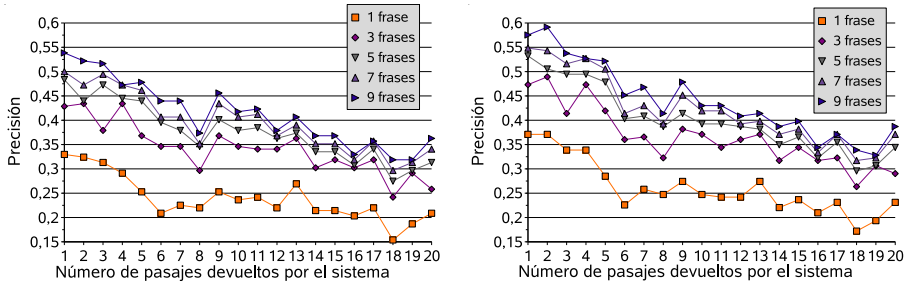


(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.23: Comparación de la precisión para diferentes tamaños de pasaje

A medida que aumentamos el tamaño del pasaje, es más probable que éste contenga la respuesta buscada y, por lo tanto, la precisión también aumenta. Así, la precisión para los pasajes de 1 frase es, aproximadamente, entre un 9 % y 12 % inferior a la misma evaluación pero con pasajes de tamaño de 3 frases. Esta diferencia se ve reducida cuando comparamos el resto de tamaños de pasajes al igual que ocurría con la cobertura, la redundancia y el MRR. Por lo demás, la curva es muy parecida independientemente del número de frases para cada pasaje.

Si observamos el precisión por posición del pasaje (figura 5.24, observamos que las diferencias de precisión se mantienen, habiendo algunos puntos en que estas diferencias se reducen considerablemente como es el caso del 16º pasaje con 3, 5, 7 y 9 frases por pasaje.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.24: Precisión por pasaje para los modelos de n-gramas con los motores SVM y RW-Denstiy

Aunque con 9 frases por pasaje se obtienen los mejores valores de cobertura, redundancia, MRR y precisión, estos pasajes son demasiado extensos para aplicarlos en sistemas completos de BR. Al ser los pasajes tan grandes, el ruido introducido no compensa la mejora de rendimiento del sistema. Por tanto es mejor escoger pasajes de tamaño de 3, 5 y 7 frases. Pero las diferencias de rendimiento de estos valores som muy similares, tanto como para declinarnos, finalmente, por el sistema con pasajes de 3 frases. El rendimiento del sistema con pasajes de 1 frase es sustancialmente inferior como para que no compense el aumento del tamaño del pasaje.

5.4. Comparación con otros sistemas de RP

Una vez encontrado los mejores parámetros de funcionamiento de los distintos modelos de n-gramas es necesario compararlos con otros sistemas para evaluar si, efectivamente, los modelos de n-gramas suponen una mejora en los sistemas de RP cuando estos se orientan a BR.

5.4.1. Descripción de los sistemas de RP

Los sistemas considerados son: IBM, que usa un modelo probabilístico basado en el bien conocido Okapi; Lucene, el sistema más utilizado por los participantes del CLEF en las tareas de BR en los últimos años; MITRE, el sistema de RI desarrollado por *The MITRE Corporation*; y el MULTITEXT que ha obtenido los mejores resultados en la comparación realizada por [TKL+03].

Lucene

Lucene es un motor de búsqueda escrito completamente en JAVA de alto rendimiento, escalable y de código abierto. Está disponible como parte del proyecto Apache Jakarta[4]. Este sistema mide la similitud entre la consulta y el documento usando el producto escalar y la función *tf/idf* como medida del peso de los términos pero Lucene promueve los documentos o pasajes más largos.

MITRE

Algoritmo basado en solapamiento de palabras desarrollado por [LMRB01]. Simplemente cuenta el número de términos de un pasaje que tiene en común con la pregunta. Los pasajes también dependen de su tamaño: el peso de cada pasaje es multiplicado por el logaritmo del número de carácteres en el pasaje que no sean espacios en blanco.

MULTITEXT

El algoritmo del MULTITEXT ([CCKL00]) es un algoritmo de recuperación de pasajes basado en densidad de palabras que favorece los

[4] http://jakarta.apache.org

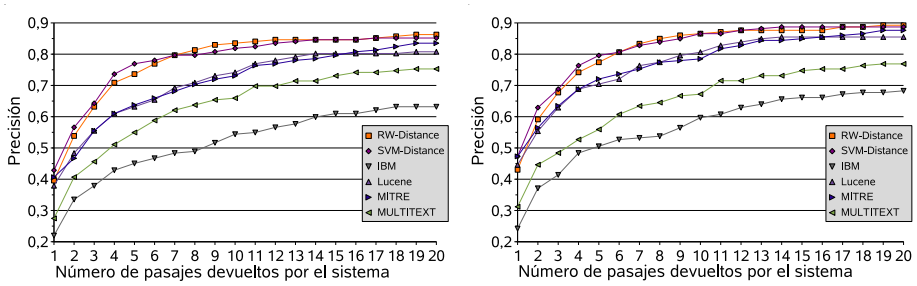
pasajes cortos que contienen muchos términos con alto valor de *idf*. También usa un etiquetador POS para identificar los verbos de la pregunta, que son buscados por su stemmer. Cada ventana del pasaje en el algoritmo empieza y termina con un término de la pregunta, y su valor de similitud está basado en el número de términos de la consulta que aparecen en el pasaje así como del tamaño de la ventana. Aunque, finalmente, el sistema devuelve el pasaje formado por tres frases enteras. Para el pesado de los términos se ha usado también el *tf/idf* estándar.

5.4.2. Evaluación de los sistemas de RP

Al igual que durante el ajuste de parámetros, en la comparación de los sistemas se va a evaluar la cobertura, redundancia, MRR y precisión de cada uno de los sistemas comentados en el párrafo anterior y compararlos con los dos mejores sistemas de n-gramas obtenidos en la sección 5.3: el RW-Distance y el SVM-Distance.

También evaluaremos el rendimiento de los sistemas para cada tipo de pregunta para observar qué sistemas trabajan mejor dependiendo del tipo de pregunta realizada.

Así, para empezar, la figura 5.25 muestra la comparativa de cobertura para los distintos sistemas evaluados.



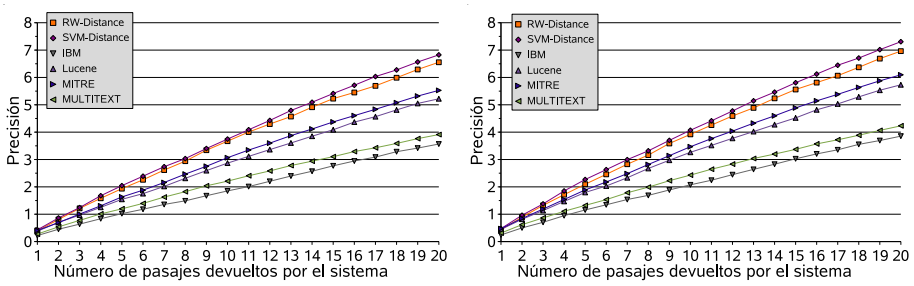
(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.25: Evaluación de la cobertura de los sistemas RW-Distance, SVM-Distance, IBM, Lucene, MITRE y MULTITEXT

En la gráfica de cobertura se observa que los sistemas con mejor rendimiento a lo largo de los primeros 20 pasajes son el RW-Distance

y SVM-Distance, los cuales tienen un comportamiento muy similar. En la evaluación estricta, donde es más difícil encontrar respuestas, los sistemas basados en n-gramas pueden superar en 13 % a sus inmediatos seguidores: Lucene y MITRE. Las diferencias se ven reducidas tanto en el primer pasaje como en el último, pero entre el 3º y 11º pasaje los resultados son destacadamente mejores. Los sistemas Lucene y MITRE tienen un comportamiento muy similar, aunque éste último obtiene un 3 % más de cobertura en el 20º pasaje que el Lucene, tanto en la evaluación estricta como indulgente. El sistema MULTITEXT, por su parte, tiene una cobertura que oscila entre un 5 % y un 14 % más baja que el MITRE, aunque lo normal sea que se encuentre entre un 6 % y un 7 % en la evaluación estricta y entre un 10 % y un 11 % en la evaluación indulgente. Pero el que ofrece peor cobertura es el sistema de IBM basado en modelos probabilísticos que puede alcanzar los 31 puntos por debajo de los modelos de n-gramas.

La redundancia de los distintos sistemas se muestra en la figura 5.26.



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.26: Comparación de la redundancia entre los distintos sistemas

Aunque la curva cobertura del RW-Distance y SVM-Distance son tan similares que no permiten distinguir qué sistema finalmente es el mejor, la curva de redundancia nos indica que el SVM-Distance, por lo menos, ofrece un poco más de redundancia que el RW-Distance. Esto quiere decir que el SVM-Distance encuentra más respuestas a una misma pregunta aunque la cobertura sea tan similar. Pero la diferencia de redundancia para estos dos sistemas es muy pequeña, siendo el SVM-Distance mejor en el 20º sólo en un 0,27 para la evaluación estricta, y 0,25 para la evaluación

indulgente. Las diferencias con el MITRE son mayores, entre 1,29 para la evaluación estricta y 1,21 para la evaluación indulgente en el 20º pasaje. El resto de sistemas dan una redundancia peor, aunque la del Lucene es muy similar a la del MITRE. Los peores sistemas son el MULTITEXT y el IBM aunque éste último vuelve a ser ligeramente peor. Con este ultimo, se encuentra más de 3 respuestas menos por pregunta que con el SVM-Distance.

El MRR de estos sistemas, como aparece en la tabla 5.14, viene a destacar los anteriores resultados.

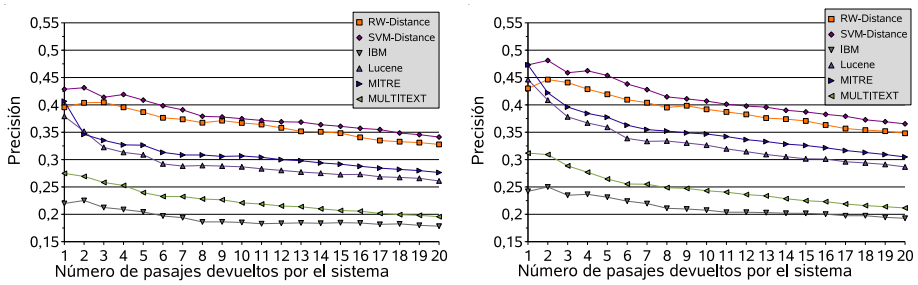
	Evaluación	
Sistema	Estricta	Indulgente
RW-Distance	0,52	0,56
SVM-Distance	0,55	0,60
IBM	0,31	0,34
Lucene	0,47	0,54
MITRE	0,49	0,56
MULTITEXT	0,38	0,41

Tabla 5.14: Comparativa del MRR en el 5º pasaje para cada uno de los sistemas evaluados

En estos resultados, vemos que el SVM-Distance vuelve a ser el mejor sistema, mejorando en 0,03 en la evaluación estricta y en 0,04 en la evaluación indulgente el MRR del RW-Distance. El siguiente que da muy buen MRR en los primeros 5 pasajes es el MITRE, que en la evaluación indulgente iguala al RW-Distance pero que en la evaluación estricta se queda en 0,03 por debajo de éste. El Lucene es ligeramente inferior al MITRE, unos 0,02 por debajo, seguido, ya a más distancia, del MULTITEXT y después del IBM. Entre el IBM y el mejor sistema (el SVM-Distance) hay una diferencia de 0,24 para la evaluación estricta y 0,26 para la evaluación indulgente.

La gráfica de precisión (figura 5.27) da resultados similares, volviendo a ser el SVM-Distance el que mejores resultados da a lo largo de toda la gráfica.

El MITRE, al igual que con la cobertura, se iguala al SVM-Distance en el primer pasaje de la evaluación indulgente, pero enseguida descien-



(a) Evaluación estricta (b) Evaluación indulgente

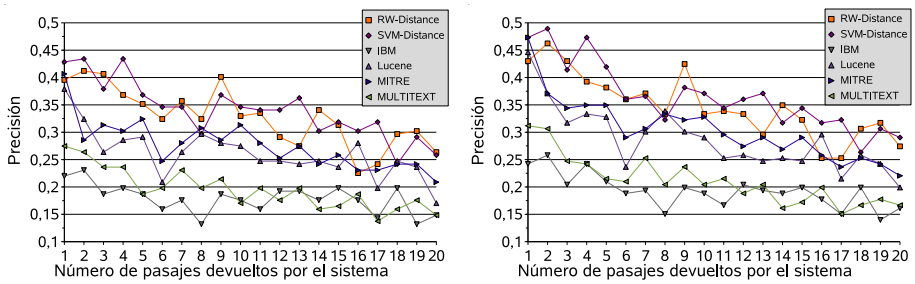
Figura 5.27: Comparación de la precisión para los distintos sistemas

de hasta situarse entre el RW-Distance y el Lucene. El RW-Distance, aunque no tiene una precisión tan buena como el SVM-Distance y el MITRE en el primer pasaje, en el segudno se acerca al SVM y mantiene, aproximadamente, su distancia a lo largo de toda la curva, reduciendo su distancia en algunos tramos. El Lucene vuelve a ser el cuarto, un poco por debajo del MITRE. Ya a más distancia se sitúa el MULTITEXT y, finalmente, y con la peor precisión, el IBM.

Usando las evaluaciones de cobertura, redundancia, MRR y precisión, podemos decir que el SVM-Distance es el sistema que mejor se comporta en todas ellas, seguido del RW-Distance. El MITRE tiene un comportamiento también muy bueno, sobretodo en el primer pasaje, pero enseguida se pone a la altura del Lucene, que tiene un rendimiento ligeramente peor. Los sistemas MULTITEXT e IBM, a pesar de haber dado buenos resultados en tareas tradicionales de RI, no son muy buenos cuando queremos obtener pasajes con la respuesta, en vez de pasajes relevantes.

Pero, ¿el SVM-Distance es el que mejor precisión obtiene en cada uno de los pasajes? La figura 5.28 nos responde a esta cuestión.

En esta figura vemos que los sistemas RW-Distance o SVM-Distance son los que mejor precisión obtienen en cada uno de los pasajes devueltos. Las únicas excepciones son cuando evaluamos la precisión del primer pasaje, en tal caso el MITRE iguala la precisión del SVM-Distance; y cuando evaluamos la precisión en el octavo pasaje, donde el Lucene supera al resto de sistemas. Cuando nos movemos a la otra parte de la gráficas, vemos que los sistemas MULTITEXT e IBM se turnan por ob-



(a) Evaluación estricta (b) Evaluación indulgente

Figura 5.28: Precisión por pasaje para los distintos sistemas evaluados

tener las peores precisiones, aunque es éste último el que obtiene peores resultados en el mayor número de pasajes.

Para descubrir qué sistemas funcionan mejor dependiendo del tipo de pregunta, debemos estudiar la tabla 5.15.

En esta tabla vemos que para las preguntas del tipo DATE, el SVM-Distance no es el mejor. El sistema MITRE es el que mejor resultados nos da para este tipo de preguntas. Obteniendo un 62 % de cobertura en el 20º pasaje en la evaluación estricta y un 75 % en la evaluación indulgente. Pero el SVM-Distance es el mejor para este tipo de preguntas cuando se examinan pocos pasajes, entre 5 y 10, alcanzando un 44 % y 50 % en los primeros 5 pasajes con las evaluaciones estricta e indulgente respectivamente. El IBM, para este tipo de preguntas, es el peor, obteniendo apenas un 25 % de las respuestas en los primeros 20 pasajes en la evaluación estricta.

Para las preguntas del tipo DEFINITION, los modelos RW-Distance y SVM-Distance son los que mejor cobertura dan, siendo el SVM-Distance mejor en los primeros pasajes y el RW-Distance en el resto. El MULTITEXT, en este caso, también da muy buenos resultados. Aunque flojea mucho en los primeros pasajes, en el 20º pasaje obtiene un 90 % y un 92 % de las respuestas para las evaluación estrictas e indulgente respectivamente. El sistema IBM vuelve a ser el peor.

En las preguntas del tipo GENERAL, para preguntas que no han podido clasificarse en ninguna de las otras caterorías, el SVM-Distance es el que mejor cobertura aporta para cualquier número de pasajes devueltos. Llegando a alcanzar un 80 % en la evaluación estricta y 90 % en

	Modelo	Número de pasajes evaluados							
	5	10	15	20		10	15	20	
DATE (19 p)	RW-Distance	0,31	**0,50**	0,50	0,50	0,44	0,62	0,62	0,62
	SVM-Distance	**0,44**	**0,50**	0,50	0,50	**0,50**	**0,69**	0,69	0,69
	IBM	0,00	0,06	0,25	0,25	0,19	0,25	0,50	0,50
	Lucene	0,38	0,44	0,50	0,56	0,44	0,56	0,69	0,69
	MITRE	0,31	0,44	**0,56**	**0,62**	**0,50**	0,56	**0,75**	**0,75**
	MULTITEXT	0,38	0,38	0,38	0,44	0,38	0,44	0,44	0,44
DEFINITION (51 p)	RW-Distance	0,82	**0,92**	**0,94**	**0,96**	0,84	**0,92**	**0,94**	**0,96**
	SVM-Distance	**0,88**	0,90	**0,94**	0,94	**0,90**	**0,92**	**0,94**	0,94
	IBM	0,50	0,64	0,76	0,78	0,55	0,69	0,80	0,80
	Lucene	0,64	0,78	0,88	0,88	0,73	0,84	0,92	0,92
	MITRE	0,58	0,74	0,86	0,90	0,71	0,78	0,90	0,92
	MULTITEXT	0,66	0,86	0,90	0,94	0,67	0,84	0,88	0,92
GENERAL (13 p)	RW-Distance	0,44	0,56	**0,67**	**0,78**	0,70	0,80	**0,90**	**0,90**
	SVM-Distance	**0,56**	**0,67**	**0,67**	**0,78**	**0,80**	**0,90**	**0,90**	**0,90**
	IBM	0,33	0,33	0,33	0,33	0,50	0,60	0,60	0,60
	Lucene	0,33	0,44	0,56	0,56	**0,80**	0,80	0,80	0,80
	MITRE	0,44	0,44	**0,67**	0,67	0,70	0,80	0,80	0,80
	MULTITEXT	0,33	0,44	0,56	0,56	0,40	0,60	0,70	0,70
NAME (93 p)	RW-Distance	**0,86**	**0,94**	0,94	**0,95**	**0,87**	**0,94**	0,94	**0,97**
	SVM-Distance	**0,86**	0,92	**0,95**	**0,95**	0,86	0,93	**0,97**	**0,97**
	IBM	0,55	0,65	0,67	0,71	0,57	0,67	0,69	0,74
	Lucene	0,77	0,85	0,90	0,90	0,82	0,87	0,91	0,91
	MITRE	0,80	0,86	0,88	0,92	0,84	0,87	0,91	0,94
	MULTITEXT	0,62	0,70	0,79	0,79	0,63	0,71	0,82	0,83
QUANTITY (24 p)	RW-Distance	**0,48**	**0,57**	**0,57**	**0,57**	**0,50**	**0,64**	**0,64**	**0,64**
	SVM-Distance	**0,48**	0,52	0,52	0,52	**0,50**	0,59	0,59	0,59
	IBM	0,33	0,33	0,38	0,38	0,36	0,36	0,36	0,36
	Lucene	0,38	**0,57**	**0,57**	**0,57**	0,36	**0,64**	**0,64**	**0,64**
	MITRE	0,43	0,52	0,52	**0,57**	0,45	0,59	0,59	**0,64**
	MULTITEXT	0,24	0,33	0,43	0,48	0,23	0,32	0,41	0,45
		Estricta				Indulgente			

Tabla 5.15: Cobertura por tipo de respuesta esperado para los sistemas comparados

la indulgente. El RW-Distance, aunque no da tan buenos resultados en los primeros pasajes, sí que los da, igualando a los del SVM-Distance, a partir del 15º pasaje. El siguiente sistema, el MITRE, está más de 10 % por debajo para este tipo de preguntas.

Los sistemas RW-Distance y SVM-Distance vuelve a ser los que mejor cobertura aportan para las preguntas del tipo NAME. Alcanzando, para ambos sistemas, porcentajes de 95 % y 97 % para las evaluaciones estricta e indulgente respectivamente. El MITRE los sigue a una distancia de entre 3 % y 7 % según el número de pasajes que se observen. El Lucene, para este tipo de preguntas, da resultados muy parecidos a los del MITRE. Y ya, a mayor distance se encuentra los sistemas MULTITEXT e IBM.

Con respecto a las preguntas del tipo QUANTITY, el RW-Distance es el que mejores resultados da a lo largo de todos los pasajes, superando al SVM-Distance en 5 % a partir de los 10 primeros pasajes. El Lucene también se comporta muy bien con este tipo de preguntas, igualando al RW-Distance a partir del 10º pasaje tanto en la evaluación estricta como indulgente. EL MITRE sólo consigue igualar los mejores resultados en los últimos 5 pasajes, siendo 5 % inferior al RW-Distance en el resto de pasajes. El sistema IBM vuelve a ser el peor para este tipo de preguntas.

En este apartado hemos visto como los sistemas RW-Distance y SVM-Distance, basados en los modelos de n-gramas, mejoran al resto de sistemas del estado del arte que hemos evaluado en cobertura, redundancia, MRR y precisión cuando aplicamos estos sistemas a la obtención de pasajes con la respuesta en vez de pasajes relevantes a una consulta. Esto nos indica que los modelos de n-gramas están mejor adaptados para incluirlos en sistemas de BR en vez de clásicos sistemas de RI basado en palabras claves. Pero no sólo mejoran los resultados cuando devuelven un número elevado de pasajes, sino que, además, lo hacen prácticamente desde el primer pasaje, obteniendo los mejores resultados a lo largo de los primeros 20 pasajes con alguna esporádica salvedad. También hemos visto que los modelos de n-gramas son mejores en cada uno de los pasajes evaluado y para los distintos tipos de preguntas excepto las de tipo DATE, las cuales el MITRE a demostrado ser el mejor cuando evaluamos un número elevado de pasajes.

5.5. Resultados en Búsqueda de Respuestas

En las dos últimas ediciones del CLEF hemos querido demostrar que JIRS mejora los resultados finales de los sistemas de BR que lo utilizan. Para ello, en la edición del 2005 utilizamos JIRS en 3 sistemas de BR distintos desarrollados por 2 grupos de investigación. Uno de estos sistemas fue desarrollado por el Laboratorio de Tecnologías del Lenguaje del Instituto Nacional de Astrofísica Óptica y Electrónica de México (INAOE) [yGVPPC+06]; otro fue desarrollado por el Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia (UPV) [GBBA+06]; y el último, fue un sistema desarrollado conjuntamente por ambos grupos [yGVPPC+05]. La único que tenían estos tres sistemas en común era que todos incorporaban, como sistema de RP, el sistema JIRS basado en alguna de sus variantes de n-gramas. Así, el grupo del INAOE utilizó el modelo SVM-Simple y el grupo UPV y grupo conjunto de UPV-INAOE utilizaron el SVM-TermWeight. En esta edición se participó en las tareas monolingües del español, francés e italiano y en la tareas bilingües de español-inglés (es decir, las preguntas en español y la colección de documentos en inglés) y de inglés-español (es decir, las preguntas en inglés y la colección de documentos en español).

Los resultados mostrados en este apartado, son los resultados ofrecidos por el CLEF del 2005 [VMG+06] y 2006 [MGF+06] en las tareas de BR donde JIRS fue usado por alguno de los sistemas participantes. Estos resultados estan representados en tablas con los siguientes campos:

run El identificador de la ejecución. Cada grupo puede enviar dos ejecuciones distintas con diferentes sistemas o parámetros. Este identificador identifica cada una de estas ejecuciones. Las ejecuciones que utilizaron JIRS como sistema de RP se han marcado en negrita.

Porcentaje de respuestas acertadas (% Right) Este porcentaje se calcula a partir del número de respuestas acertadas dividido por el número total de preguntas que tienen respuesta en la colección de documentos.

Número total de respuestas incorrectas (W) El número de respuestas que el sistema ha contestado incorrectamente.

Número de respuestas inexactas (X) Junto a la respuesta, los sistemas entregan un documento del cual han obtenido la información. Los evaluadores del CLEF comprueban si esa respuesta, efectivamente, aparece en el documento y que ese documento justifica esa respuesta. Las respuestas no soportadas son aquellas que el documento asociado a la respuesta en realidad no responde a la pregunta aunque la respuesta sea correcta.

Porcentaje de respuestas del tipo factual (% F) Porcentaje de las respuestas factuales correctamente respondidas.

Porcentaje de respuestas del tipo definición (% D) Porcentaje de respuestas del tipo definición correctamente respondidas.

Porcentaje de respuestas del tipo temporal (% T) Porcentaje de respuestas cuyas preguntas tienen algún tipo de restricción temporal.

Precisión, Recall y F-medida de las respuestas NIL (P, R y F) Las respuestas NIL son aquellas cuya respuesta no está soportada por la colección de documentos, es decir, que no existe ningún documento en la colección que dé respuesta a esa pregunta. Estos valores calculan la precisión, recall y f-medida de los sistemas en detectar este tipo de preguntas.

Confidence-weight Score (CWS) Junto con la respuesta el identificador del documento que la soporta, los sistemas deben devolver un valor que indique en qué grado dicho sistema confía en la respuesta. El CWS es un coeficiente que determina en qué medida este valor de confianza es fiable, en otras palabras, si el sistema, cuando se equivocó, dio un grado de confianza pequeño y cuando acertó grande.

En la tabla 5.16 podemos ver los resultados de los participantes de la tarea monolingüe en español.

Para esta tarea el grupo de la UPV mandó dos ejecuciones (*upv051* y *upv052*). La segunda ejecución era idéntica a la primera excepto que hacía uso de la Web para validar la respuesta. En el momento de generar los resultados para el CLEF tuvimos un problema con la conexión a

run	Right %	W #	X #	U #	Right %F [118]	Right %D [50]	Right %T [32]	NIL[20] P	NIL[20] R	NIL[20] F	CWS
inao051	42.00	110	5	1	28.81	80.00	31.25	0.23	0.80	0.36	-
tova051	41.00	109	7	2	28.81	80.00	25.00	0.24	0.55	0.33	-
inao052	39.50	116	4	1	27.12	80.00	21.88	0.19	0.80	0.31	-
tova052	38.50	113	8	2	23.73	80.00	28.12	0.22	0.55	0.32	-
upv051	33.50	119	13	1	26.27	52.00	31.25	0.19	0.30	0.23	0.218
alia051	33.00	110	24	-	29.66	40.00	34.38	0.25	0.45	0.32	0.170
aliv051	32.50	116	18	1	28.81	46.00	25.00	0.26	0.25	0.26	0.15
alia052	30.00	114	26	-	26.27	36.00	34.38	0.24	0.45	0.32	0.153
talp051	29.00	122	20	-	27.97	36.00	21.88	0.26	0.70	0.38	0.089
talp052	27.00	133	13	-	25.42	32.00	25.00	0.22	0.65	0.33	0.078
mira051	25.50	138	11	-	26.27	34.00	9.38	0.08	0.10	0.09	0.123
mira052	23.00	140	14	-	22.03	34.00	9.38	0.08	0.10	0.09	0.103
upv052	18.00	155	9	-	22.88	0.00	28.12	0.10	0.40	0.16	0.128

Tabla 5.16: Resultados de la tarea de BR monolingüe en español

Internet que no pudimos subsanar a tiempo y los resultados para dicha ejecución se vieron muy afectados por ello. Por otra parte, el resto de ejecuciones que utilizaron JIRS como sistema de RP (marcadas en negrita en la tabla), obtuvieron las primeras posiciones en cuanto a porcentaje de respuestas acertadas. Los mejores resultados fueron obtenidos para las preguntas del tipo definición, con una precisión de un 80 % para el grupo del INAOE y el grupo combinado INAOE-UPV, el grupo de la UPV en solitario alcanzó el 52 %. Cabe destacar el comportamiento del sistema *alia051* del grupo de Alicante que, pese no haber alcanzado los mayores valores, sí son los que menos respuestas totalmente incorrectas respondieron, teniendo 24 respuestas parcialmente respondidas o no del todo exáctas. Además, son el grupo con más respuestas acertadas del tipo factual (el grupo más numeroso). Por otra parte, el coeficiente CWS del sistema de la UPV basado en n-gramas es el más alto. Para calcular este factor, nuestro sistema de BR se basaba, casi integramente, en el valor de similitud del primer pasaje que devolvía JIRS en que se creía que contenía la respuesta. En otras palabras, el valor de similitud que

run	Right %	W #	X #	U #	%F [120]	Right %D [50]	%T [30]	NIL[20] P	R	F	CWS
syna051	64.00	62	8	2	59.17	86.00	46.67	0.23	0.25	0.24	-
tova052	35.00	120	10	-	27.50	66.00	13.33	0.14	0.30	0.19	-
tova051	34.50	121	10	-	26.67	66.00	13.33	0.13	0.25	0.17	-
upv051	23.00	143	7	4	17.50	46.00	6.67	0.06	0.10	0.07	0.115
hels051	17.50	156	8	1	16.67	22.00	13.33	0.10	0.45	0.17	0.108
upv052	17.00	160	5	1	15.00	20.00	20.00	0.07	0.20	0.10	0.073
lire051	16.50	145	20	2	15.83	14.00	23.33	0.09	-	0.09	0.072
hels052	16.50	157	10	-	15.00	22.00	13.33	0.09	0.40	0.15	0.097
lina051	14.50	144	21	3	17.95	6.00	16.67	0.15	0.20	0.17	0.048
lcea051	14.00	165	3	4	18.33	0.00	20.00	0.33	0.05	0.09	-

Tabla 5.17: Resultados de la tarea de BR monolingüe en francés

da JIRS junto con los resultados, indica en qué grado es sencillo o difícil extraer la respuesta de los pasajes y lo hace con mayor precisión que el resto de sistemas. El grupo del INAOE e INAOE-UPV no entregaron este valor y, por lo tanto, no se han podido evaluar el CWS de estos sistemas.

En la tabla 5.17 podemos apreciar los resultados del CLEF en la tarea monolingüe en francés.

Los resultados para el francés, aunque no obtuvieron los mejores resultados, los sistemas que usaron JIRS como sistema de RP estuvieron entre las primeras posiciones. La primera posición se la llevó la compañía francesa Synapse Développment al utilizar una compleja ontología que dió también muy buenos resultados con el portugués [VMG+06]. En estas ejecuciones no fueron tan buenos resultados como en el caso del español puesto que los patrones utilizados para el módulo de extracción de la respuesta no eran tan refinados. Así, mientras que en español, con la ejecución conjunta *tova051* alcanzamos un 41 % y un 33.50 % con la ejecución *upv051*, en francés sólo alcanzamos un 35 % con *tova052* y un 23 % con *upv051*. En estos resultados no tuvimos el mismo problema que con en el español al usar la Web pero el resultado, en nuestra ejecución *upv052*, empeoró la precisión un 6 % pese a que las preguntas con restricciones temporales (T) aumentase un 7 %. De nuevo, nuestro valor de

confianza en la respuesta (CWS), como media, fue el mejor, aunque los sistemas *tova* y *syna* no aportaron este valor.

Con los resultados en italiano volvimos a obtener los mejores resultados, como se puede apreciar en la tabla 5.18.

	Right	**W**	**X**	**U**		**Right**		**NIL**[20]			
run	%	#	#	#	**%F** [120]	**%D** [50]	**%T** [30]	**P**	**R**	**F**	**CWS**
tova052	27.50	135	10	-	23.33	42.00	20.00	0.15	0.55	0.24	-
tova051	26.50	138	9	-	21.67	42.00	20.00	0.16	0.55	0.24	-
upv051	25.50	142	6	1	20.00	44.00	16.67	0.10	0.15	0.12	0.156
upv052	24.00	148	4	-	15.83	50.00	13.33	0.06	0.15	0.09	0.125
irst051	22.00	137	17	2	19.17	38.00	6.67	0.17	0.20	0.18	0.129
irst052	19.00	145	14	3	14.17	38.00	6.67	0.40	0.10	0.16	0.100

Tabla 5.18: Resultados de la tarea de BR monolingüe en italiano

Aunque en esta ocasión únicamente tuvimos que enfrentarnos al grupo de investigación ITC-Irst. En italiano empeoran los resultados de las ejecuciones *tova051* y *tova052* con respecto al francés, pero mejoran las ejecuciones *upv051* y *upv052* pese a que no alcanzan los valores obtenidos para el español. Las diferencias entre la ejecución *upv052*, con validación Web, y *upv051*, sin ningún tipo de validación de la respuesta, se reducen, aunque este primero sigue siendo ligeramente peor y, al contrario que ocurría con el francés, las preguntas con restricciones temporales no mejoran, sino las de definición.

En las tareas bilingües existe la dificultad añadida de, o bien se traduce la pregunta, o bien se traduce la colección de documentos. En ambos casos se cometen muchos errores de traducción aunque, en un principio, la traducción de la pregunta parece una tarea más abordable. Nosotros hemos obtado por este método utilizando, para ello, sistemas de traducción on-line. En la tabla 5.19 se muestran los resultados obtenidos.

En este tarea sólo se presentó otro grupo a parte del nuestro con dos ejecuciones. Aquí, en la ejecución que enviamos al CLEF, el resultado es casi idéntico al que obtuvimos con el francés. Teniendo en cuenta que en la tarea bilingüe es más complicado obtener buenos resultados dado los errores cometidos en la traducción de la pregunta de forma

run	Right %	W #	X #	U #	Right %F [118]	%D [50]	%T [32]	NIL[20] P	R	F	CWS
upv051	22.50	139	14	2	19.49	34.00	15.62	0.15	0.20	0.17	0.103
mira052	19.50	151	8	2	16.95	28.00	15.62	0.17	0.25	0.20	0.088
mira051	19.50	153	7	1	16.95	28.00	15.62	0.17	0.25	0.20	0.093

Tabla 5.19: Resultados de la tarea de BR bilingüe con preguntas en inglés y la colección de documentos en español

automática, nuestro sistema de BR para el francés no fue muy bueno. Dado que la independencia del idioma de JIRS quedó patente en los experimentos realizados en [GyGS+05], esto indica que las reglas que usamos para extraer la respuesta en la colección de documentos en francés no fueron muy acertadas. Así, nuestro sistema superó a las dos ejecuciones restantes (*mira051* y *mira052*) con un 22.50 % de precisión. Obtuvimos mejor porcentaje en las preguntas factuales y de definición y exáctamente el mismo en las preguntas con restricciones temporales, aunque no fuimos capaces de detectar las respuestas no soportandas tan bien. Otra vez, nuestro sistema fue mejor en el coeficiente de confianza (CWS).

Aunque fuimos los únicos que se presentaron a la tarea de BR bilingüe español-inglés y no podemos compararlos con otros competidores, los resultados de la tabla 5.20 nos puede indicar varias cosas.

run	Right %	W #	X #	U #	Right %F [120]	%D [50]	%T [30]	NIL[20] P	R	F	CWS
upv051	17.00	156	9	1	12.40	28.00	17.24	0.15	0.50	0.23	0.072

Tabla 5.20: Resultados de la tarea de BR bilingüe con preguntas en español y la colección de documentos en inglés

Los resultados en esta tarea son los peores debido a la dificultad de traducir correctamente las preguntas, pero también nos indica que se realizó un esfuerzo mayor en obtener las reglas de extacción de la respuesta para el español ya que, los resultados donde la colección de documentos

estaba en español los resultados fueron mejores que sus homólogos en otros idiomas. Así, nuestro sistema de BR fue 5.5 % mejor en la tarea inglés-español que español-inglés. Solamente las preguntas con restricciones temporales fueron 1.62 % mejores.

En el 2006 apenas se hicieron cambio en el sistema de BR que presentamos al CLEF. Se mejoró ligeramente las reglas y se cambio el sistema RW-TermWeight que usamos en el 2005 por el RW-Distance. Pero nuestro principal objetivo fue demostrar que JIRS es un sistema de RP que obtiene mejores resultados en BR que el sistema más utilizado por los participantes del CLEF: Lucene. Para ello lanzamos dos ejecuciones exactamente con los mismos parámetros pero cambiando el motor de búsqueda, en otras palabras, una ejecución usaría el RW-Distance y la otra el Lucene como sistema de RP. En la tabla 5.21 se puede ver esta comparativa para los idiomas español, francés e italiano.

Idioma	**JIRS**				**Lucene**			
	% Right	**% F**	**% T**	**% D**	**% Right**	**% F**	**% T**	**% D**
Español	**35,00**	**37,04**	25,00	**47,63**	28,50	27,78	25,00	40,48
Francés	**31,58**	**31,08**	-	**33,33**	24,74	26,35	-	19,05
Italiano	28,19	**28,47**	-	26,83	28,19	27.78	-	**29,27**

Tabla 5.21: Resultados en las tareas monolingües en español, francés e italiano

Tanto en español como en francés el sistema que utiliza JIRS como RP devuelve un 6 % más de precisión en encontrar la respuesta. Esto no ocurre con el italiano debido a que su corpus es más pequeño que los corpus de español y francés. Esto parece indicar que JIRS funciona mejor cuando hay alta redundancia y esto se consigue en colecciones de documentos grandes, en otro caso se comporta como los sistemas de RP basado en palabras claves.

Capítulo 6
Conclusiones

Los actuales sistemas de RI usados en las más prestigiosas campañas de evaluación de sistemas de BR multilingües no son apropiados para la tarea. Esto es debido a que los sistemas o bien son tradicionales sistemas de RI mal adaptados para trabajar con sistemas de BR o bien son altamente dependientes del idioma. El problema de los primeros es que su objetivo es devolver pasajes con mayor relevancia dada una consulta hecha por el usuario en vez de devolver pasajes con mayor probabilidad de contener la respuesta a una pregunta. Los segundos son muy difíciles de adaptar a nuevos idiomas o a ambientes multilingües, requiriendo en muchos casos volver a implementar gran parte de la aplicación para cada nuevo idioma o realizar una ardua tarea de escritura y adaptación de reglas de conocimiento.

La falta de cobertura, precisión y redundancia de los actuales sistemas de RP, establece limitaciones muy serias al rendimiento máximo que los sistemas de BR pueden alcazar. Debido a que los sistemas de RI o RP son un primer filtro a la información que el sistema de BR procesará, es importante que sean capaces de devolver fragmentos de texto con la respuesta. Es imposible que el resto del sistema de BR pueda responder a la pregunta sino se le entrega texto con la respuesta.

JIRS es un sistema de RI de gran flexibilidad y escalabilidad. Permite modificar tanto la configuración como el comportamiento de todo el sistema cambiando unos pocos parámetros; añadir o modificar operatividad al sistema sin necesidad de modificar el código existente; ejecutar la aplicación localmente, con arquitectura cliente/servidora o de forma distribuida; y permite crear nuevas aplicaciones que, automáticamente, adquieran las características de JIRS: configuración, escalabilidad, ejecu-

ción local y remota, flexibilidad, etc.

JIRS utiliza modelos especializados de RP que están especialmente adaptados para tareas de BR. Para ello se aprovecha de la redundancia de las colecciones de documentos para obtener los pasajes con mayor probabilidad de contener la respuesta y que, además, sea más fácil de extraer. Pero en el caso de corpora con poca redundancia, JIRS iguala su rendimiento a otros sistemas de RP. Estos modelos han sido adaptados a idiomas como español, inglés, francés, italiano, árabe, urdu y oromo, y han sido evaluados en cinco de éstos con excelentes resultados.

Los resultados presentados en la presente tesis muestran que JIRS obtiene un mayor rendimiento cuando se aplica a sistemas de BR que otros sistemas del estado del arte como el IBM, Lucene, MITRE y MULTITEXT superando, en ocasiones, en 13 puntos a su inmediato seguidor. Pero no sólo mejora el rendimiento de todo el sistema, sino que lo hace para todos los tipos de preguntas evaluados y desde el primer pasaje devuelto.

Además, los modelos de n-gramas de JIRS han dado muy buenos resultados en las campañas de evaluación del CLEF en los años 2005 y 2006. En el primer año de participación, todos los sistemas de BR que incorporaron JIRS como sistema de RP alcanzaron las mejores precisiones. En la última edición del CLEF en la que participamos se realizó una comparativa entre el Lucene, que es el sistema de RI más utilizado por los participantes del CLEF, y el modelo de n-gramas de Densidad de Distancias del JIRS, alcanzando una mejora de un 6 % de cobertura en español y francés.

JIRS mejora la cobertura, precisión y redundancia de la respuesta en cualquier idioma en los que se aplique siempre y cuando el corpus tenga suficiente tamaño y, por lo tanto, bastante redundancia. En caso contrario JIRS actúa como los tradicionales sistemas de RP.

Entre las mejoras del sistema nos proponemos, como trabajos futuros, la siguiente lista:

- Especializar las búsquedas para cada tipo de pregunta. Los resultados han sido muy prometedores siendo un sistema independiente del idioma. Por ello pretendemos mejorar la cobertura reduciendo esta independiencia. Esto lo conseguiremos añadiendo búsquedas especializadas que dependan del tipo de respuesta esperado y del idioma de la colección.

- Aplicar filtros a los pasajes devueltos por el sistema. Para algunos tipos de preguntas, como las del tipo fecha o cantidad, se espera, como respuesta, un tipo de datos que es muy sencillo de detectar. Por ello, y aprovechano esta característica, sería interesante filtrar los pasajes devueltos por el sistema de tal manera que eliminen aquellos pasajes que no contengan datos para estos tipos de preguntas. Por ejemplo, si la pregunta es de tipo cantidad, eliminar los pasajes que no contengan números.

- Reformular las preguntas para hacer más probable que su estructura aparezca junto con la respuesta. Hay ciertos tipos de respuesta que no suelen aparecer con una sintaxis similar a la expresión utilizada en la pregunta. También hay preguntas que son demasiado largas o que el usuario las realiza con un particular estilo poco común. Para estas preguntas es necesario reformular la pregunta, o extender la pregunta usando algún tipo de realimentación o thesaurus. Creemos que en los casos de extensión de la pregunta, JIRS, por sus características, será capaz de fomentar aquellas palabras de la extensión que aparezcan junto con el resto de los términos de la pregunta que sí aparecen en la colección y que discriminará aquellas que no, al ser poco probables de ser encontradas en el mismo contexto.

- Expansión de fechas implícitas usando la metainformación de los documentos. En muchas preguntas del tipo fecha, el sistema encuentra expresiones del tipo "*hoy*", "*mañana*" o "*próximo martes*" que no han sido tomadas como respuestas válidas y, por lo tanto, queremos que el sistema sea capaz de reconocer estas estructuras y cambiarlas por las fechas exactas teniendo en cuenta la metainformación del documento al que pertenece la noticia.

- Mejorar la redundancia de los corpora utilizando colecciones de documentos en varios idiomas. JIRS mejora cuando la redundancia es elevada, por ello queremos realizar experimentos en los que las preguntas están en un idioma dado y las colecciones de documentos sean colecciones de periódicos del mismo año aunque éstos estén en varios idiomas. Con estos experimentos comprobaremos si el incremento de redundancia es suficiente para compensar los errores

de traducción.

- Una de las hipótesis de JIRS es que fomenta aquellas estructuras sintáctica y semánticamente correctas debido a que, las que no lo son, no aparecen en la colección de documentos. Queremos comprobar si esta hipótesis es cierta aplicando JIRS a la salida de traductores para escoger, combinar y mejorar las traducciones automáticas. También vemos la posibilidad de utilizar JIRS como correctores ortográficos, sintácticos y semánticos buscando expresiones similares en grandes colecciones de documentos para saber en qué grado esas expresiones son utilizadas en su contexto.

- Internet es, hoy por hoy, la colección de documentos más grande y con mayor redundancia. Por ello queremos probar los modelos de n-gramas de JIRS con información procedente de Internet. Para ello pretendemos adaptar el motor de búsqueda para que trabaje con los buscadores Web más usados.

- Aunque hemos creado tres modelos de n-gramas para evaluar JIRS, únicamente utilizamos una función de pesado de términos, otra de n-gramas y otra de distancias, así pretendemos integrar y probar nuevas funciones de pesado y de cálculo de distancias para términos y n-gramas.

- Por último nos proponemos incrementar las conjuntos de preguntas y respuestas añadiendo las preguntas del CLEF de otros años y otros idiomas para mejorar la precisión de la evaluaciones y comprobar que el sistema se sigue comportando igual al cambiar las colecciones.

Hasta el momento, los resultados han sido muy prometedores obteniendo muy buenos resultados con nuestras evaluaciones y las realizadas por los evaluadores del CLEF. Pero pretendemos mejorar el sistema hasta alcanzar una cobertura de la respuesta de hasta el 90 % en los primeros 5 pasajes devueltos por el sistema.

Las características de JIRS lo hacen una aplicación idónea para la investigación en áreas de la RI dada su elevada modularidad, flexibilidad y escalabilidad. Es un proyecto OpenSource que pretende facilitar la investigación a innumerables científicos que trabajen en esta área. Por ello

nos hemos esforzado en crear una herramienta sencilla pero potentente y de fácil configuración en la que puerdan participar en su creación grupos de todo el mundo sin invertir excesivo tiempo en su aprendizaje. También hemos pretendido construir una aplicación útil para usuarios que busquen algo más que una lista de documentos relevantes a una consulta. Pretendemos que sea una herramienta de uso común para usuarios que busquen respuestas entre los miles de millones de documentos que pueden alcanzar desde su ordenador.

Bibliografía

[AAB+05] Risuh Ahn, Beatrix Alex, Johan Bos, Tiphaine Dalmas, Jochen L. Leidner, and Matthew B. Smillie. Cross-lingual question answering using off-the-shelf machine translation. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491 of *Lecture Notes in Computer Science*, pages 446–457, Bath, UK, 2005. Springer Berlin / Heidelberg.

[ACC+00] James Allan, Margaret E. Connel, W. Bruce Croft, Fang-Fang Feng, David Fisher, and Xiaoyan Li. Inquery and trec-9. In *Proceedings of Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards & Technology, 2000.

[AFM+06] Carlos Amaral, Helena Figueira, André Martins, Alfonso Mendes, Pedro Mendes, and Claudia Pinto. Priberams question answering system for portuguese. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 410–419, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[AKM05] Lili Aunimo, Reeta Kuuskoski, and Juha Makkonen. Finnish as source language in bilingual question answering. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491 of *Lecture Notes in Computer Scien-*

ce, pages 482–493, Bath, UK, 2005. Springer Berlin / Heidelberg.

[AR06] Mirna Adriani and Rinawati. Finding answers to indonesian questions from english documents. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 510–516, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[BCC+02] John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, and Ellen Voorhees. Issues, tasks, and program structures to roadmap research in question & answering (q&a). In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 535–537, New York, NY, USA, 2002. ACM Press.

[BD01] Sabine Buchholz and Walter Daelemans. Shapaqa: Shallow parsing for question answering on the world wide web. In *Proceedings Euroconference Recent Advances in Natural Language Processing*, pages 47–51. RANLP, 2001.

[BDB02] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, pages 257–264. Association for Computational Linguistics, 2002.

[BEF06] Romaric Besançon, Mehdi Embarek, and Olivier Ferret. The œdipe system at clef-qa 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 337–346, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[BK81] Duncan A. Buell and Donald H. Kraft. Threshold values and boolean retrieval systems. *Information Processing and Management*, 3(17):127–136, 1981.

[BLB+01] Eric Brill, Jimmy Lin, Michele Banko, Susan T. Dumais, and Andrew Y. Ng. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*. National Institute of Standards & Technology, 2001.

[BMvN+06] Gosse Bouma, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jorg Tiedemann. Question answering for dutch using dependency relations. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 370–379, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[Boo80] A. Bookstein. Fuzzy request: An approach to weighted boolean searches. *Journal of the American Society for Information Science*, 4(31):240–247, July 1980.

[Boo81] A. Bookstein. A comparison of two systems of weighted boolean retrieval. *Journal of the American Society for Information Science*, 4(32):275–279, July 1981.

[Buc01] Sabine Buchholz. Using grammatical relations, answer frequencies and the world wide web for trec question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*. National Institute of Standards & Technology, 2001.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Wokingham, UK, 1999.

[Cal94] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual ACM SIGIR Conference on Research and Development in Information*

Retrieval, Annual ACM Conference on Research and Development in Information Retrieval, pages 302–310, Dublin, Ireland, 1994. Springer-Verlag.

[CCKL00] Charles L. A. Clarke, Gordon V. Cormack, Derek I. E. Kisman, and Thomas R. Lynam. Question answering by passage selection (multitext experiments for trec-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards & Technology, 2000.

[CCL01] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 358–365, New York, NY, USA, 2001. ACM Press.

[CCPK99] Gordon V. Cormack, Charles L. A. Clarke, Christopher R. Palmer, and Derek I. E. Kisman. Fast automatic passage ranking (multitext experiments for trec-8). In *Proceedings of Eighth Text REtrieval Conference (TREC-8)*, pages 735–742. National Institute of Standards & Technology, 1999.

[CLMS+00] Jim Cowie, Evgeny Ludovik, Hugo Molina-Salgado, Sergei Nirenburg, and Svetlana Scheremetyeva. Automatic question answering. In *Proceedings of the RIAO Conference*, Paris, France, 2000. RIAO.

[Cos06] Luís Costa. 20th century esfinge (sphinx) solving the riddles at clef 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 467–476, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[CTCTC04] Kevyn Collins-Thompson, Jamie Callan, Egidio Terra, and Charles L. A. Clarke. The effect of document retrieval quality on factoid question answering performance.

In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 574–575, Sheffield, United Kingdom, 2004. ACM Press.

[DCEyGVP04] Alejandro Del-Castillo-Escobedo, Manuel Montes y Gómez, and Luis Villaseñor-Pineda. Qa on the web: a preliminary study for spanish language. In *Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04)*, pages 322–329, Colima, Mexico, 2004. IEEE Computer Society.

[DYC03] Florence Duclaye, Francois Yvon, and Olivier Collin. Learning paraphrases to improve a question-answering system. In *Proceedings of 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, 2003. Association for Computational Linguistics.

[FKG+06] Daniel Ferrés, Samir Kanaan, Edgar González, Alicia Ageno, Horacio Rodríguez, and Jordi Turmo. The talp-qa system for spanish at clef 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 467–476, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[FKK+99] Michael Fuller, Marcin Kaszkiel, Sam Kimberley, Corinna Ng, Ross Wilkinson, Mingfang Wu, and Justin Zobel. The rmitcsiro ad hoc, qa, web, interactive, and speech experiments at trec 8. In *Proceedings of Eighth Text REtrieval Conference (TREC-8)*, pages 549–564. National Institute of Standards & Technology, 1999.

[GBBA+06] José Manuel Gómez, Davide Buscaldi, Empar Bisbal-Asensi, Paolo Rosso, and Emilio Sanchis. *QUASAR: The Question Answering System of the Universidad Politécnica de Valencia*, volume 4022 of *Lecture Notes in Compu-*

ter Science, pages 439–448. Springer Berlin / Heidelberg, Viena, Austria, 2006.

[GGH+03] Robert Gaizauskas, Mark A. Greenwood, Mark Hepple, Ian Roberts, Horacio Saggion, and Matthew Sargaison. The university of sheffield's trec 2003 q&a experiments. In *Proceedings of the Tweltieth Text REtrieval Conference (TREC-12)*. National Institute of Standards & Technology, 2003.

[GH00] Robert Gaizauskas and Kevin Humphreys. A combined ir/nlp approach to question answering against large text collections. In *Proceedings of RIAO 2000: Content-Based Multimedia Information Access*, pages 1288–1304, Paris, France, April 2000.

[GIM+05] Brigitte Grau, Gabriel Illouz, Laura Monceaux, Isabelle Robba, Anne Vilnatand Guillaume Bourdil, Faïza Elkateb-Gara, Olivier Ferret, and Benoît Mathieu. Answering french questions in english by exploiting results from several sources of information. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491 of *Lecture Notes in Computer Science*, pages 470–481, Bath, UK, 2005. Springer Berlin / Heidelberg.

[Gre04] Mark A. Greenwood. Using pertainyms to improve passage retrieval for questions requesting information about a location. In *Proceedings of the Workshop on Information Retrieval for Question Answering (SIGIR 2004)*, Sheffield, UK, 2004.

[GyGS+05] José Manuel Gómez, Manuel Montes y Gómez, Emilio Sanchis, Luis Villaseñor-Pineda, and Paolo Rosso. Language independent passage retrieval for question answering. In *Fourth Mexican International Conference on Artificial Intelligence MICAI 2005*, Lecture Notes in Computer Science, pages 816–823, Monterrey, Mexico, 2005. Springer Verlag.

[Har88] Donna Harman. Towards interactive query expansion. In *Proceedings of the Eleventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 321–331, Grenoble, France, 1988. ACM Press.

[Har92a] Donna Harman. Relevance feedback and other query modifcation techniques. In *Information Retrieval: Data Structures and Algorithms*, pages 241–263, New York, USA, 1992. Prentice Hall, Inc.

[Har92b] Donna Harman. Relevance feedback revisited. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 1–10, Copenhagen, Denmark, 1992. ACM Press.

[Har92c] Donna Harman. User-friendly systems instead of user-friendly front-ends. *Journal of the American Society for Information Science*, 43(2):164–174, 1992.

[Har06] Sven Hartrumpf. Extending knowledge and deepening linguistic processing for the question answering system insicht. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 361–369, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[HEM02] Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. Natural language based reformulation resource and web exploitation for question answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*. national Institute of Standars & Technology, 2002.

[HGH+00] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in web-

clopedia. In *Proceeding of the Ninth Text REtrieval Conference (TREC-9)*. National Institute of Standards & Technology, 2000.

[HP93] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 59–68, Pittsburgh, Pennsylvania, USA, 1993. ACM Press.

[HTC98] David Hawking, Paul Thistlewaite, and Nick Crasswell. Anu/acsys trec-6 experiments. In *Proceding of the Sixth Text REtrieval Conference (TREC-6)*, pages 275–290. National Institute of Standars & Technology, 1998.

[IFZR00] Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. Ibm's statistical question answering system. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 229–234. National Institute of Standards & Technology, 2000.

[JJJW99] Pierre Jourlin, Sue E. Johnson, Karen Spärck Jones, and Philip C. Woodland. General query expansion techniques for spoken document retrieval. In *Proceedings of the ESCA ETRW Workshop: Accessing Information in Spoken Audio*, pages 8–13, 1999.

[JL02] Hanmin Jung and Gary Geunbae Lee. Multilingual question answering with high portability on. In *Procedings of the 2002 conference on Multilingual Summarization and Question Answering*, volume 19 of *International Conference On Computational Linguistics*, pages 1–8. Association for Computational Linguistics, 2002.

[JvRA+06] Valentin Jijkoun, Joris van Rantwijk, David Ahn, Erik Tjong Kim Sang, and Maarten de Rijke. The university of amsterdam at clef@qa 2006. In *Working notes of Cross*

Language Evaluation Forum 2006, Alicante, Spain, 2006. CLEF.

[KZ97] Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 178–185, Philadelphia, Pennsylvania, USA, 1997. ACM Press.

[LC02] Xiaoyong Liu and W. Bruce Croft. Passage retrieval based on language models. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, Conference on Information and Knowledge Management, pages 375–382, McLean, Virginia, USA, 2002. ACM Press.

[Lit99] Kenneth C. Litkowski. Question-answering using semantic relation triples. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*. National Institute of Standards & Technology, 1999.

[LMRB01] Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7(4):325–342, December 2001.

[LP01] Fernando Llopis-Pascual. *IR-N: Un Sistema de Recuperación de Información Basado en Pasajes*. Phd. thesis, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante, Alicante, Spain, 2001.

[LSL+01] Gary Geunbae Lee, Jungyun Seo, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Chanhki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, Harksoo Kim, and Kyungsun Kim. Siteq: Engineering high performance qa system using lexico-semantic pattern matching and shallow nlp. In *Proceedings of the Tenth*

Text REtrieval Conference (TREC-10), pages 442–451. National Institute of Standards & Technology, 2001.

[LSN06] Dominique Laurent, Patrick Séguéla, and Sophie Nègre. Cross-lingual question answering using qristal for clef 2006. In *In Working notes of Cross Language Evaluation Forum 2006*, Alicante, Spain, 2006.

[LVF02] Fernando Llopis, José Luis Vicedo, and Antonio Ferrández. Passage selection to improve question answering. In *Proceedings of the COLING 2002 Workshop on Multilingual Summarization and Question Answering*, volume 19 of *International Conference On Computational Linguistics*, pages 1–6, Taipei, Taiwan, 2002. Association for Computational Linguistics.

[MBF+90] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on wordnet. Technical Report 43, Cognitive Science Laboratory, 1990.

[MDVFCS05] Enrique Méndez-Díaz, Jesús Vilares-Ferro, and David Cabero-Souto. Cole experiments at qa@clef 2004 spanish monolingual track. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491 of *Lecture Notes in Computer Science*, pages 544–551, Bath, UK, 2005. Springer Berlin / Heidelberg.

[MGF+06] Bernardo Magnini, Danilo Giampiccolo, Pamela Forner, Christelle Ayache, Valentin Jijkoun, Petya Osenova, Anselmo Peñas, Paulo Rocha, Bogdan Sacaleanu, and Richard Sutcliffe. Overview of the clef 2006 multilingual question answering track. In *Working notes of Cross Language Evaluation Forum 2006*, Alicante, Spain, 2006.

[MNPT02] Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. Multilingual question/answering: the diogene system. In *NIST special publication SP*, volume 250, pages 313–321. National Institute of Standards & Technology, 2002.

[Mon03] Christof Monz. Document retrieval in the context of question answering. In *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 571–579, New York, USA, 2003. Springer Berlin / Heidelberg.

[MP06] Jolanta Mizera-Pietrasko. Accuracy of the aid system's information retrieval in processing huge data collections. In *In Working notes of Cross Language Evaluation Forum 2006*, Alicante, Spain, 2006.

[MPHS03] Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, volume 21 of *ACM Transactions on Information Systems (TOIS)*, pages 327–335, New York, USA, 2003. ACM Press.

[NIH+00] Isao Namba, Nobuyuki Igata, Hisayuki Horai, Kiyoshi Nitta, and Kunio Matsui. Fujitsu laboratories trec9 report. In *Proceedings of the Nineth Text REtrieval Conference (TREC-9)*. National Institute of Standards & Technology, 2000.

[NS05] Günter Neumann and Bogdan Sacaleanu. Experiments on robust nl question interpretation and multi-layered document annotation for a crosslanguage question/answering system. In *Multilingual Information Access for Text, Speech and Images: 5th Workshop of the Cross-Language Evaluation Forum*, volume 3491 of *Lecture Notes in Computer Science*, pages 411–422, Bath, UK, 2005. Springer Berlin / Heidelberg.

[OSS+05] Petya Osenova, Alexander Simov, Kiril Simov, Hristo Tanev, and Milen Kouylekov. Bulgarian-english question answering: Adaptation of language resources. In *Multilingual Information Access for Text, Speech and Images:*

5th Workshop of the Cross-Language Evaluation Forum, volume 3491 of *Lecture Notes in Computer Science*, pages 458–469, Bath, UK, 2005. Springer Berlin / Heidelberg.

[PSGLMF+06] César Pablo-Sánchez, Ana González-Ledesma, José Luis Martínez-Fernández, José Maria Guirao, Paloma Martinez, and Antonio Moreno. Miracles cross-lingual question answering experiments with spanish as a target language. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 488–491, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[QF93] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *Proceedings of the 16th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 160–169, Pittsburgh, Pennsylvania, United States, 1993. ACM Press.

[RFF+06] S. Roger, S. Ferrández, A. Ferrández, J. Peral, F. Llopis, A. Aguilar, and D. Tomás. Aliqan, spanish qa system at clef-2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 447–466, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[RFQ+02] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering on the web. In *Proceedings of the 11th international conference on World Wide Web*, International World Wide Web Conference, pages 408–419, Honolulu, Hawaii, USA, 2002. ACM Press.

[RG04] Ian Roberts and Robert J. Gaizauskas. Evaluating passage retrieval approaches for question answering. In *26th*

European Conference on IR Research, ECIR 2004, volume 2997 of *Lecture Notes in Computer Science*, pages 72–84, Pisa, Italy, 2004. Springer Berlin / Heidelberg.

[RSJ88] Stephen E. Robertson and Karen Sparck-Jones. Relevance weighting of search terms. *Document Retrieval Systems*, 27:143–160, 1988.

[Sal71] Gerard Salton. *The SMART Retrieval SystemExperiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[Sal89] Gerald Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Series In Computer Science. Addison-Wesley Longman Publishing Co., Inc., Ithaca, NY, 1989.

[SB87] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. Technical report, Cornell University, Ithaca, NY, USA, 1987.

[SFW83] Gerad Salton, Edward A. Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26:1022–1036, November 1983.

[Sme97] Alan F. Smeaton. Information retrieval: Still butting heads with natural language processing? In *International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, Lecture Notes In Computer Science, pages 115–138, London, UK, 1997. Springer-Verlag.

[SMG+06] Richard F. E. Sutcliffe, Michael Mulcahy, Igal Gabbay, Aoife OGorman, Kieran White, and Darina Slattery. Cross-language french-english question answering using the dlt system at clef 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the*

Cross-Language Evaluation Forum, volume 4022 of *Lecture Notes in Computer Science*, pages 502–509, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[SMS06] Robert Strötgen, Thomas Mandl, and René Schneider. A fast forward approach to cross-lingual question answering for english and german. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 332–336, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[SN06] Bogdan Sacaleanu and Günter Neumann. Multiple language question answering track. In *In Working notes of Cross Language Evaluation Forum 2006*, Alicante, Spain, 2006.

[TKL+03] Stefanie Tellex, Boris Katz, Jimmy J. Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 41–47, Toronto, Canada, 2003. ACM Press.

[TKM+06] Hristo Tanev, Milen Kouylekov, Bernardo Magnini, Matteo Negri, and Kiril Simov. Exploiting linguistic indices and syntactic structures for multilingual question answering: Itc-irst at clef 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 390–399, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[TVSI06] David Tomás, José L. Vicedo, Maximiliano Saiz, and Rubén Izquierdo. An xml-based system for spanish question answering. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation*

Forum, volume 4022 of *Lecture Notes in Computer Science*, pages 347–350, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[VILM03] José L. Vicedo, Ruben Izquierdo, Fernando Llopis, and Rafael Muñoz. Question answering in spanish. In *Comparative Evaluation of Multilingual Information Access Systems: 4th Workshop of the Cross-Language Evaluation Forum, CLEF 2003*, volume 3237 of *Lecture Notes in Computer Science*, pages 541–548, Trondheim, Norway, 2003. Springer Berlin / Heidelberg.

[VMG+06] Alessandro Vallin, Bernardo Magnini, Danilo Giampiccolo, Lili Aunimo, Christelle Ayache, Petya Osenova, Anselmo Peñas, Maarten de Rijke, Bogdan Sacaleanu, Diana Santos, and Richard Sutcliffe. Overview of the clef 2005 multilingual. question answering track. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evaluation Forum*, volume 4022 of *Lecture Notes in Computer Science*, pages 307–331, Viena, Austria, 2006. Springer Berlin / Heidelberg.

[Voo94] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 61–69, Dublin, Ireland, 1994. Springer-Verlag.

[Voo99] Ellen M. Voorhees. The trec-8 question answering track report. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 77–82. National Institute of Standards & Technology, 1999.

[WBM94] Ian H. Witten, Timothy C. Bell, and Alistair Moffat. *Managing Gigabytes: Compressing and Indexing Documents and Images, 1st edition.* John Wiley & Sons, Inc., New York, NY, USA, 1994.

[Wil94] Ross Wilkinson. Effective retrieval of structured documents. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 311–317, Dublin, Ireland, 1994. Springer-Verlag.

[WK79] W. G. Waller and D. H. Kraft. A mathematical model for a weighted boolean retrieval system. *Information Processing and Management*, 15:235–245, 1979.

[XC96] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the nineth annual international ACM SIGIR conference on Research and development in information retrieval*, Annual ACM Conference on Research and Development in Information Retrieval, pages 4–11, Zurich, Switzerland, 1996. ACM Press.

[yGVPPC+05] Manuel Montes y Gómez, Luis Villaseñor-Pineda, Manuel Pérez-Coutiño, José M. Gómez, Emilio Sanchis, and Paolo Rosso. Inaoe-upv joint participation in clef 2005: Experiments in monolingual question answering. In *Working Notes for the CLEF 2005 Workshop*, Vienna, Austria, 2005. CLEF.

[yGVPPC+06] Manuel Montes y Gómez, Luis Villaseñor-Pineda, Manuel Pérez-Coutiño, José Manuel Gómez-Soriano, Emilio Sanchis-Arnal, and Paolo Rosso. *A Full Data-Driven System for Multiple Language Question Answering*, volume 4022 of *Lecture Notes in Computer Science*, pages 420–428. Springer Berlin / Heidelberg, Viena, Austria, 2006.

[Zad65] L. A. Zadeh. Fuzzy sets. In *Readings in Fuzzy Sets for Intelligent Systems*, volume 8, pages 338–353. Morgan Kaufmann, 1965.

[Zip49] George Kingsley Zipf. *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, 1949.

[ZMWSD95] Justi Zobel, Alistair Moffat, Ross Wilkinson, and Ross Sacks-Davis. Efficient retrieval of partial documents. In *Information Processing and Management*, volume 31, pages 361–377, 1995.

Apéndice A
Glosario de abreviaturas

BE Booleano Extendido

BR Búsqueda de Respuestas

CLEF Cross Language Evaluation Forum

EI Extracción de la Información

EV Espacio Vectorial

JDBIR JAVA DataBase Information Retrieval

JIRS JAVA Information Retrieval System

JPM JAVA Process Manager

MRR Media Reciprocal Rank

PLN Procesamiento del Lenguaje Natural

RD Recuperación de Documentos

RI Recuperación de Información

RP Recuperación de Pasajes.

tf/idf Term-frequency/inverse document frequency

TREC Text REtrieval Conference

Apéndice B
Manual de usuario

JAVA Information Retrieval System (JIRS) es una aplicación muy potente y con un alto grado de adaptabilidad. Profundizando un poco en su funcionamiento se puede realizar muchas más cosas para las cuales no estaba diseñado inicialmente gracias a su núcleo JAVA Process Mananager (JPM). Pero en este capítulo me centraré exclusivamente en el uso básico de JIRS como herramienta para indexar y buscar documentos y pasajes.

B.1. Instalación

La instalación de la aplicación se realiza mediante los siguientes pasos:

1. Descargar la aplicación de alguno de estos dos enlaces:

```
http://jirs.dsic.upv.es/downloads/jirs2.3.0_jdk1.5.tgz
http://jirs.dsic.upv.es/downloads/jirs2.3.0_jdk1.5.zip
```

   Dependiendo si deseas la aplicación comprimida en formato *zip* o empaquetada con el formato *tgz*.

2. Descomprimir la aplicación en cualquier directorio.

3. Y comprobar que se accede a la Máquina Virtual de JAVA mediante el comando:

```
java -version
```

Para la versión de JIRS que cuelga en los anlaces anteriores es recomendable que la versión de JAVA sea la 1.5.x. De todas formas se incluyen los fuentes de la aplicación para el que necesite trabajar con la versión 1.4.x de JAVA pueda compilarlo para ésta.

B.2. Modo de uso

Como acabo de explicar, JIRS está constituido por su núcleo JPM. Así, para lanzar la aplicación se debe utilizar la siguiente orden:

```
jpm configuration [run_name [process_name]]
        [-parameter_name parameter_value]...
jpm -h
```

Donde:

configuration Es un archivo de configuración de ejecuciones (RUNS) escrito en XML donde se especifica las acciones y parámetros por defecto del sistema. Junto a la aplicación se adjunta un fichero de configuración llamado 'runs.xml' y tiene todo lo necesario para indexar y buscar documentos y pasajes. El presente apéndice explica las diferentes ejecuciones que se puede realizar con este archivo de configuración.

run_name Dentro del fichero de configuración existen diferentes ejecuciones, es decir, diferentes acciones a realizar. Con este parámetro especificamos qué ejecución realizará. Si no especificamos el parámetro entonces lanzará la primera ejecución definida en el archivo de configuración.

process_name Se puede especificar con qué proceso empezará la ejecución **run_name**, si no especificamos este parámetro entonces empezará por el primer proceso de la ejecución.

parameter_name parameter_value Podemos cambiar cualquier parámetro por defecto de la aplicación especificando su nombre y su valor. Estos parámetros pasarán a ser dinámicos y por lo tanto globales al sistema.

-h Este parámetro nos visualiza una pequeña ayuda en línea como aparece en la figura B.1.

```
JAVA Process Manager version 2.2.4
Usage:
  jpm configuration [run_name [process_name]]
[-parameter_name parameter_value]...

Where:
  configuration - XML configuration file for one specific run.
  run_name - Main run name, if it omits the system executes the first run
of config file.
  process_name - Process name to execute at first place within the
execution (or run).
  parameter_name - Is a parameter of configuration name. This become in
dinamic global parameter.

This parameters are dinamics parameters and its have got priority over
the statics parameters of config file.
Example:
  jpm config.xml -buffer_length 4096 -stop_list common_words

System know parameters:
 -verbose level: print more information of run processes. 'level' can be
equal or more than 0. When more big is the level number more information
it supply.
 -noverbose level: no print more automatic verbose message of process
with more level but yes that prints the user's␣messages␣of␣the␣processes.
␣-encoding:␣file␣encoding␣(UTF-8,␣ISO8859-1,...)
␣-h␣or␣--help:␣print␣this␣help
```

Figura B.1: Ejemplo de la ayuda en línea de la aplicación jpm

B.3. Parámetros globales

Los parámetros globales son parámetros definidos globalmente al principio del archivo de RUNS. Es aconsejable que estos parámetros sean modificados y establecidos antes de cualquier ejecución. Hay dos formas de cambiar estos parámetros:

1. Modificar (preferentemente con un editor XML aunque también se puede con un editor de texto) el archivo de RUNS ('runs.xml').

Desde la línea 5 hasta la 36 se definen todos los parámetros globales del sistema como se muestra en el siguiente código:

```
 5 <!-- Global parameters -->
 6 <parameters>
 7  <!-- Default document collection language -->
 8  <parameter name="language">english</parameter>
 9  <!-- Collection name -->
10  <parameter name="collection">clefqa</parameter>
11  <!-- Path to config files -->
12  <parameter name="config_files">config_files/</parameter>
13  <!-- Path to temporal directory -->
14  <parameter name="tmp_dir">tmp/</parameter>
15  <!-- Path to indexed files -->
16  <parameter name="database_dir">indexeds/$collection$_$language$/
</parameter>
17  <!-- Size of buffer of reading and writing on disk -->
18  <parameter name="buffer_length">65536</parameter>
19  <!-- URL or IP address of the search server -->
20  <parameter name="host">localhost</parameter>
21  <!-- Port to connect to the server -->
22  <parameter name="port">8081</parameter>
23  <!-- If the run mode is graphical (graphic) o text (text) -->
24  <parameter name="mode">graphic</parameter>
25  <!--Select the replace character file deppend on the language-->
26  <switch param="language">
27    <case value="english|spanish|french|italian|german">
28      <parameter name="replace_character_file">$config_files$latin
_repl_chars.xml</parameter>
29      <parameter name="clean_text_file">$config_files$latin_clean_
text.xml</parameter>
30    </case>
31    <case value="arabic|urdu">
32      <parameter name="replace_character_file">$config_files$arabic_
repl_chars.xml</parameter>
33      <parameter name="clean_text_file">$config_files$arabic_clean_
text.xml</parameter>
34    </case>
35  </switch>
36 </parameters>
```

Así, si yo quisiera cambiar el parámetro 'language' permanentemente, tendría que cambiar en el archivo anterior el parámetro con el nombre 'language'.

Si se modifica los parámetros en el archivo de configuración de RUNS estos quedan permanentemente cambiados en futuras ejecuciones de la aplicación. Sin embargo, si lo que se pretende es cambiar un parámetro para una ejecución puntual de la aplicación es más aconsejable modificar dinámicamente los parámetros.

2. Modificar dinámicamente los parámetros del sistema. Estos permanecen cambiados durante toda la ejecución JIRS hasta que finaliza, después este parámetro vuelve a tener el valor definido en el archivo de RUNS. Para cambiar cualquier parámetro del sistema para una ejecución 'run_name' dada se utilizaría el siguiente comando:

```
jpm runs.xml run_name -parameter_name value
```

   Por ejemplo, el parámetro global *language* para la ejecución de indexado de la colección de documentos puede ser modificado de la siguiente forma:

```
jpm runs.xml Indexing -language french
```

   Con este comando estamos ejecutando jpm para que indexe una colección de documentos con el parámetro global *language* con el valor de *french*.

Aunque la lista de parámetros globales puede ser definida y cambiada en el archivo de ejecuciones RUNS, por defecto estos parámetros globales del sistema son:

language Especifica el idioma de la colección. Por defecto *english*.

collection Especifica un identificador para la colección. Por defecto *clefqa*. Este parámetro junto con el parámetro del idioma lo utilizará JIRS para generar la estructura por defecto de directorios y ficheros para la base de datos. Por ejemplo, si queremos indexar una de las colección del CLEF en español para las tareas de búsqueda de respuestas, podemos definir el identificador de la colección como 'clefqa' y el idioma 'spanish'. Esto, si no se especifica lo contrario, nos generará un directorio con el nombre 'clefqa_spanish' donde almacenará los ficheros invertidos y la base de datos de esta colección.

config_files Ruta al directorio donde se ubican más archivos de configuración. Por defecto en el subdirectorio 'config_files' del directorio actual.

tmp_dir Un directorio donde se almacenarán los archivos temporales. Por defecto en el subdirectorio *tmp* del directorio actual.

database_dir Directorio donde se almacenará los ficheros asociados a la base de datos y ficheros invertidos. Por defecto este directorio se creará en *indexeds/* a partir del nombre y el idioma de la colección como se ha explicado anteriormente. Como se puede observar en el listado de los parámetros globales, el valor del parámetro 'database_dir' se construye a partir de texto plano y del valor de otros dos parámetros 'collection' y 'language'. Así, el valor de cualquier parámetro puede ser definido a partir del valor otros parámetros poniendo el nombre del parámetro al que se hace referencia entre dos símbolos $. Si se quiere poner el símbolo $ a un parámetro se debe hacer escribiendo dos $ seguidos, por ejemplo, si quisieramos establecer la frase '*El $ aumentó su coticación en bolsa*' como valor de un parámetro dado habría que introducirla como: '*El $$ aumentó su coticación en bolsa*'.

buffer_length Tamaño del buffer de lectura y escritura en disco. Por defecto *64KB*.

host En operaciones cliente servidor y/o distribuidas puedes especificar a qué máquina se debe conectar. Por defecto *localhost*.

port Especifica el puerto de conexicón TCP/IP. Por defecto *8081*.

mode El modo gráfico en que se ejecutará la aplicación. Por defecto está en modo gráfico (*graphic*) pero cuando la aplicación se ejecuta en un entorno de texto, se debe especificar modo texto (*text*).

A parte de los parámetros globales antes mencionados existen otros parámetros globales y locales que no explicaré aquí sino en las secciones que hagan uso de ellos.

B.4. Indexar documentos

Antes de empezar a utilizar JIRS como un motor de búsqueda, es necesario indexar una o varias colecciones de documentos. Para ello será necesario:

- Una colección de documentos en formato del CLEF o TREC.

  Aunque en la siguiente versión de JIRS pretendo crear un formato propio escrito en XML, por el momento JIRS sólo soporta los formatos del CLEF o TREC. Estos documentos están escrito en SGML y, aunque poseen muchos campos, para JIRS sólo es necesario documentos con la siguiente estructura:

```
<DOC>
 <DOCNO>ID_DEL_DOCUMENTO_1</DOCNO>
 <TEXT>
TEXTO DEL DOCUMENTO 1
 </TEXT>
</DOC>
<DOC>
 <DOCNO>ID_DEL_DOCUMENTO_2</DOCNO>
 <TEXT>
TEXTO DEL DOCUMENTO 2
 </TEXT>
</DOC>
...
```

  En un archivo pueden haber tantos documentos como se quiera. También es posible pasarle varios archivos al sistema con varios documentos en cada uno de ellos. Lo que no es posible es dividir un mismo documento entre varios archivos.

- Una lista de documentos a indexar.

  JIRS es capaz de procesar varios archivos durante el proceso de indexación. Por lo tanto es necesario indicarle un archivo XML con las rutas a cada uno de los archivos que debe indexar. El archivo XML es muy sencillo y tiene la siguiente estructura:

```
<?xml version="1.0" encoding="UTF-8"?>
<file_list>
  <file>D:/Corpora/english/latimes94/la010194.sgml.gz</file>
  <file>D:/Corpora/english/latimes94/la010294.sgml.gz</file>
  <file>D:/Corpora/english/latimes94/la010394.sgml.gz</file>
  <file>D:/Corpora/english/latimes94/la010494.sgml.gz</file>
</file_list>
```

  Es decir, en cada elemento 'file' del archivo XML se debe especificar la ruta completa a uno de los archivos que se quiera indexar. Esta lista se puede generar mediante el siguiente comando de JIRS:

```
jpm runs.xml CreateDocList [-folder dir] [-pattern p] [-file f]
                           [-recursive (true|false)]
```

Donde:

folder Será el directorio desde donde empezará la búsqueda de los archivos (por defecto el actual).

pattern El patrón regular JAVA de búsqueda. Es decir, todos los ficheros que cumplan con ese patrón serán incluidos en la lista. Por defecto este patrón JAVA es '*.*\.sgml*' que corresponde al patrón '**.sgml*', es decir, todos los archivos que tengan extensión *sgml* serán incluidos en la búsqueda.

recursive Si incluirá en la búsqueda los subdirectorios. Por defecto *false*.

file Fichero donde se almacenará la lista de archivos a indexar en formato XML. Por defecto se creará un fichero XML formado por el identificador de la colección y el idioma dentro del directorio '*config_files/doc_lists/*'.

Por ejemplo:

```
jpm runs.xml CreateDocList -folder clef/english/ -recursive true
```

Este comando crea un archivo XML con el nombre de '*clefqa_doc_list-english.xml*' (si no se ha cambiado los parámetros globales *language* ni *collection*) en el directorio '*config_files/doc_lists/*' con todos los archivos con extensión '*.sgml*' que se encuentren en el directorio o subdirectorios de '*clef/english/*'.

- Una lista de palabras de comunes.

  Aunque no es obligatorio, es recomendable darle una lista de palabras comunes o *stopwords* para reducir el tamaño de los índices y ficheros invertidos[1]. Esta lista de palabras comunes debe almacenarse en un fichero XML con la siguiente sintaxis:

[1] En http://www.unine.ch/info/clef/ se puede encontrar listas de palabras para una multitud de idiomas.

```
<stopwords>
<word>a</word>
<word>a's</word>
<word>able</word>
<word>about</word>
<word>above</word>
<word>according</word>
<word>accordingly</word>
<word>across</word>
<word>actually</word>
...
</stopwords>
```

Donde cada elemento XML 'word' contiene una palabra común. JIRS aporta la lista de palabras comunes para árabe, inglés, francés, italiano, español y urdu en el directorio '*config_files/stopwords/*'.

Una vez preparada la colección de documentos, creada la lista de archivos a indexar y el archivo con las palabras comunes podemos indexar dicha colección de documentos utilizando el siguiente comando:

```
jpm runs.xml Indexing [-doc_list archivo] [-include_tags tags]
                      [-delim reg_expr] [-file_decompress_method (gz|zip|none)]
```

Donde:

doc_list La ruta al archivo XML con la lista de archivos a indexar. Su valor por defecto depende de los parámetros globales *collection* y *language* como se ha explicado anteriormente.

include_tags Los formatos del CLEF y TREC pueden incluir distintos campos (no sólo el campo *TEXT*), los únicos que son obligatorios es el *<DOCNO>*. Por lo tanto hay que especificar qué campos (separados por comas) se incluirán cuando indexemos la colección de documentos. Por defecto se han incluido los campos necesarios para las colecciones del CLEF de inglés, finlandés, francés, italiano, portugués, ruso y español.

delim Este parámetro indica una expresión regular JAVA que nos indicará los límites de los términos durante el proceso de indexación[2].

[2]Los límites de los términos durante la indexación y la búsqueda pueden ser distintos.

Por defecto, los límites de los términos es cualquier secuencia de uno o más signos de puntuación o espacios. Esta expresión debe formularse con la sintaxis de las expresiones regulares en JAVA y no se debe tocar almenos que se sepa qué se está haciendo.

file_decompress_method Especifica el tipo de descompresión que se realizará a los documentos de la colección antes de indexarlos. Existen dos métodos el *GZip* (*gz*) y el *Zip* (*zip*). Por defecto no hay ningún mecanismo de compresión.

A estos parámetros se les puede incluir los parámetros globales explicados anteriormente.

Por ejemplo, podemos indexar la colección de documentos, simplemente introduciendo:

```
jpm runs.xml Indexing
```

Esto indexará la colección de documentos con los parámetros por defecto que están almacenados en el archivo '*runs.xml*'.

Si lo que queremos es especificar una nueva lista de documentos, se podría lanzar:

```
jpm runs.xml Indexing -doc_list mi_lista.xml
```

Si nuestros ficheros están comprimidos escribiríamos uno de estos comandos:

```
jpm runs.xml Indexing -file_decompress_method gz
jpm runs.xml Indexing -file_decompress_method zip
```

Dependiendo del método de compresión que hayamos utilizado.

Para cambiar los límites de los términos, por ejemplo, sólo por espacios, podríamos lanzar el siguiente comando:

```
jpm runs.xml Indexing -delim "␣+"
```

Esto indicaría que los límites de los términos estarían delimitados por uno o más espacios.

B.5. Comprobar si la indexacción se ha realizado con éxito

Una vez efectuada la indexacción, podemos ver los índices y ficheros invertidos que se ha generado con el siguiente comando:

```
jpm runs.xml ViewDatabase
```

Este comando es el único que no tiene equivalente en modo texto (por el momento) y el parámetro global *-mode text* no funciona. Si en el proceso de indexación se han cambiado alguno de los parámetros por defecto de forma dinámica (es decir, en la línea de comandos), entonces aquí se deben introducir los mismos parámetros.

La ejecución *ViewDatabase* abre una ventana que contiene 5 pestañas con la información de las 5 tablas de índices y ficheros invertidos generadas. Con el ratón es fácil moverse entre las diferentes tablas y observar los valores. Además, se puede desplazar a cualquier registro de una tabla dada introduciendo su número en el campo de texto *Register*.

Las tablas que genera son las siguientes:

Documents Con el identificador del documento orginal (el que aparece en la etiqueta "*<DOCNO>*" en los archivos orginales en el formato del CLEF) y una clave externa a la tabla **DocumentsLocation**, donde se describe dónde está ubicado físicamente el documento en el disco duro.

DocumentLocation La ruta a los ficheros donde se encuentran los documentos indexados.

Passages La lista de pasajes extraídos de los documentos. Por cada pasaje se almacena: el identificador del documento al que pertenece el pasaje (*iddoc*); el peso del pasaje asignado durante la indexación (*weight*); la frecuencia máxima que un término aparece en el pasaje (*freqmax*); y el texto del pasaje (*text*).

TermsDocument Esta tabla relaciona los identificadores de los términos que aparecen en cada pasaje (*terms*) y su frecuencia en dicho

pasaje (*freqs*). El número de registro (*Reg*) se asocia diréctamente con el número de registro de la tabla de pasajes.

Histogram Esta tabla muestra los términos y sus atributos, como *idterm* con el identificador del término, *word* con el texto del término, *freq* con la frecuencia de aparición, *ndocs* con el número de pasajes en los que aparece el término, *docs* con la lista de identificadores de pasajes en los que aparece el término ordenados de menor a mayor y *freqs* con la lista de frecuencias en las que el término aparece en cada documento de la lista anterior.

B.6. Lanzar el motor de búsqueda de pasajes

Aunque se puede lanzar una búsqueda en modo local, es mejor realizar búsquedas en un modelo cliente/servidor (figura B.2). Esto es así puesto que, antes de empezar una búsqueda, debe cargarse en memoria los índices necesarios para que la búsqueda sea más eficiente y esto puede tardar unos minutos. Así, mientras que en modo local hay que cargar los índices cada vez que queremos lanzar la aplicación de búsqueda, en modo cliente/servidor lanzaríamos el motor de búsqueda en el servidor una única vez y después le haríamos peticiones desde el cliente todas las veces que quisieramos sin necesidad de cargar de nuevo los índices. Únicamente, cuando quisieramos cambiar los parámetros de búsqueda, tendríamos que parar el servidor y volver a arrancarlo con los nuevos parámetros.

La instrucción para lanzar el motor de búsqueda es:

```
jpm runs.xml SearchPassagesServer [-npassages num] [-filter (yes|no)]
        [-disfactor factor] [-model (distance|termweight|simple|rw|svm)]
      [-naddlines num] [-reformulation (yes|no)] [-searchengine (rw|svm)]
        [-term_weither (soft-idf|ne-soft-idf|tf-idf)] [-nse_passages num]
                                        [-combination (union|intersection)]
```

Donde:

npassages Indica cuantos pasajes debe devolver el sistema al final.

model El modelo global de búsqueda que se utilizará (por defecto *distance*). Cada modelo se iniciará con sus parámetros por defecto. Las opciones son:

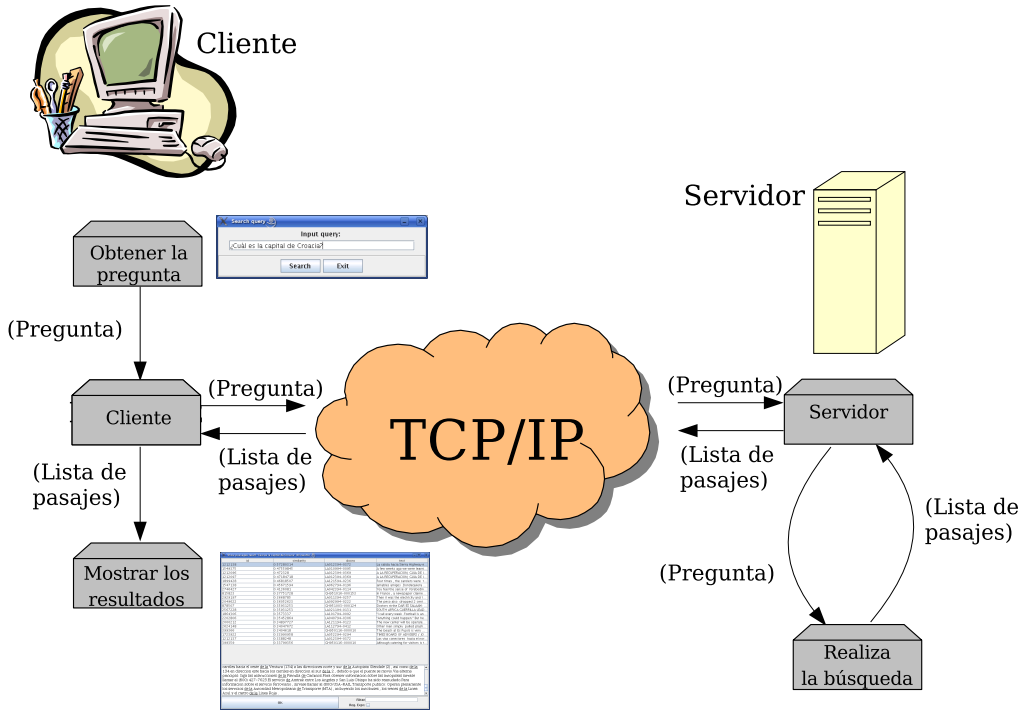


Figura B.2: Modulos que participan en la búsqueda Cliente/Servidor

svm El modelo espacio vectorial clásico, con pesado de términos *tf-idf*.

rw El modelo de palabras relevantes. El cuál sólo tiene en cuenta el peso de las palabras relevantes de la consulta. En este modelo no tiene en cuenta el peso de los documentos ni el de la propia consulta. Para el pesado de términos utiliza una aproximación basada en el *tf-idf*.

simple El modelo de n-gramas simple.

termweight El modelo de n-gramas de pesado de términos.

distance El modelo de n-gramas de densidad de distancias.

filter Si este parámetro está en *yes* el sistema filtrará, de la salida, los pasajes que no cumplan una ciertas condiciones, como que no estén

presentes las palabras pivote. El sistema considera palabras pivote aquellas que empiecen por mayúscula o número, por ejemplo, *Croacia* o *1987*. También filtra por tipo de respuesta esperado, por ejemplo, si el tipo esperado es un número, filtra aquellos pasajes que no aparezca ninguna cantidad, ya sea escrita con número o con letra. Los tipos que reconoce son:

DATE.YEAR Un número que representa el año. Es decir, una cifra que vaya desde 1200 hasta 2999.

DATE Filtra aquellos pasajes que no contenga ningún mes del año o un año.

DATE.DAY Sólo mantiene aquellos pasajes que aparece el día, el més y el año a la vez y con la fórma estándar de cada idioma.

QUANTITY.MONEY Filtra los pasajes que no contienen alguna de estas monedas: dólar, euro, ecu, lira, franco, marco, libra, peseta, peso, rublo y yen.

QUANTITY Cantidades en general, tanto en letras como en cifras.

QUANTITY.DIMENSION Unidades de medida: metro, kilo, kilómetro, hectárea y litro.

QUANTITY.AGE Filtra aquellos pasajes que no tienen alguna de estas palabras: edad, año, siglo, decenio y milenio.

Estos filtros sólo funcionan para el inglés, español, francés e italiano, aunque fácilmente se podrían añadir más filtros para otros idiomas.

JIRS no tiene un módulo de reconocimiento del tipo de pregunta, el tipo de pregunta debe ser introducido por el usuario. En el apartado B.9 se explica cómo se puede asignar tipos de respuesta a las consultas. En el caso de que la pregunta no contenga ningún atributo con el tipo de la respuesta esperado, entonces no aplica el filtro.

naddlines Establece el tamaño del pasaje indicando cuantas frases se añadirán a la frase central del pasaje. Es decir, si este parámetro vale 0, el sistema devolverá como pasaje únicamente la frase que

contenga las palabras buscadas; si vale 1, a la frase se le añadirán la frase anterior y posterior; si vale 2, a la frase central se le añadirán dos frases anteriores y dos posteriores y así sucesivamente. Hay que tener en cuenta que cuanto mayor es el pasaje mayor es la probabilidad de encontrar la respuesta pero también el sistema requiere más recursos.

disfactor El factor de distancia para el modelo *distance*. Determina en qué grado la distancia entre n-gramas es importante. Cuando menor es el número menos importante es la distancia. Si este valor es 0, entonces no se tiene en cuenta la distancia y da igual que los n-gramas estén separados o juntos. Por defecto su valor es de 0.1.

reformulation Si se aplica la reformulación de la pregunta para dos tipos de preguntas: DEFINITION.ORGANIZATION (para preguntas del tipo definición de organización) y DEFINITION.PERSON (para definición de personas).

searchengine Antes de aplicar los modelos de n-gramas se debe utilizar un motor de búsqueda. En esta versión de JIRS hay dos posibilidades, el estándar Space Vectorial Model (svm) o Relevant Words (rw). Éste último diseñado especialmente para su funcionamiento con los modelos de n-gramas. Por defecto sólo el modelo *svm* tiene el motor de búsqueda *svm*, el resto tiene el motor *rw* que ha demostrado dar muy buenos resultados en los experimentos y consume menos recursos que el *svm*.

term_weighter Qué función de pesado se aplicará sobre los términos.

nse_passages Número de pasajes que devolverá el motor de búsqueda. Debido a los recursos necesarios por los modelos de n-gramas es necesario limitar el número de pasajes devueltos por el motor de búsqueda. Este límite es, por defecto, 1000.

combination Indica cómo se combinan los diferentes pasajes que se encuentran por cada uno de los términos. Si se aplica la *union* (que es el valor por defecto), entonces la lista final de pasajes será la unión de éstos, si no será la intersección.

B.7. Realizar una búsqueda de pasajes remota

Una vez lanzado el motor de búsqueda se debe iniciar un cliente para que le haga peticiones al servidor y presente los resultados. La sintaxis es:

```
jpm runs.xml SearchPassages [-client/server yes] [-fields fields]
                                                 [-repeat (yes|no)]
```

Donde:

client/server Por defecto es *yes* y determina que las peticiones al motor de búsqueda las haga en modo cliente/servidor.

fields Determina qué campos se visualizarán en modo texto, por defecto son: *id*, *similarity*, *docno* y *text*.

repeat Determina si pregunta al usuario si se realiza otra consulta cuando termina una. Si la opción es *no* entonces el programa termina después de realizar la búsqueda.

B.8. Realizar una búsqueda de forma local

Esto permite realizar una búsqueda sin una estructura cliente/servidor. La sintaxis es una unión del servidor y del cliente de búsqueda:

```
jpm runs.xml SearchPassagesServer -client/server no [-filter (yes|no)]
          [-model (distance|termweight|simple|rw|svm)] [-naddlines num]
   [-disfactor factor] [-reformulation (yes|no)] [-searchengine (rw|svm)]
       [-term_weither (soft-idf|ne-soft-idf|tf-idf)] [-nse_passages num]
    [-combination (union|intersection)] [-npassages num] [-fields fields]
                                                      [repeat (yes|no)]
```

Para una descripción detallada de los argumentos, mirar los apartados B.6 y B.7.

B.9. Buscar una colección de preguntas automáticamente

Hay ocasiones que, en vez de realizar una pregunta del usuario al sistema, se desea lanzar una colección de preguntas previamente almacenadas, con el objetivo de evaluar los resultados de forma estadística y hacer calculos de precisión y cobertura del sistema. Para ello JIRS tiene una ejecución específica para ello.

Al igual que para hacer una pregunta en modo cliente/servidor, el primer paso es arrancar el servidor como se explica en la sección B.6 con los parámetros que se deseen. Después se puede realizar peticiones al servidor a partir de una colección de preguntas realizada en dos formatos: el formato del CLEF o un formato própio basado en XML.

B.9.1. Formato de preguntas del CLEF

Si se decide utilizar el formato del CLEF, el archivo deberá ser un archivo de texto con la siguiente sintaxis:

```
TYPE ID_QUESTION FROM TO QUESTION
TYPE ID_QUESTION FROM TO QUESTION
...
```

Cada línea representa una pregunta y cada campo está separado por uno o más caracteres en blanco (incluyendo el tabulador). Donde:

TYPE El tipo de la pregunta. Al contrario del formato original de CLEF, este campo no tiene por qué ser un único carácter.

ID_QUESTION Identificador de cada pregunta, este indicador puede ser cualquier combinación de números y letras.

FROM Identificador del idioma de la pregunta. Por ejemplo: "ES".

TO Identificador del idioma de la colección de documentos.

QUESTION Pregunta. Esta pregunta puede ser tan larga como se desee y utilizar espacios y carácteres de interrogación, comillas, etc.

Un ejemplo en el formato original del CLEF:

```
D 0001 ES ES What is BMW?
D 0002 ES ES What are the FARC?
D 0003 ES ES What is Nelson Mandela?
D 0004 ES ES What is Javier Solana?
...
```

B.9.2. Formato XML de JIRS (JIRS Questions)

Para aumentar la portabilidad entre sistemas, se ha desarrollado un formato de preguntas en formato XML. Su formato es:

```
<questions>
 <question id="0001">
   <atributes>
     <atribute name="name1">value_atribute_1</atribute>
     <atribute name="name2">value_atribute_2</atribute>
     ...
     <atribute name="nameN">value_atribute_N</atribute>
   </atributes>
   <langstring lang="query_lang">query_1</langstring>
 </question>
 <question id="0002">
   <atributes>
     <atribute name="name1">value_atribute_1</atribute>
     <atribute name="name2">value_atribute_2</atribute>
     ...
     <atribute name="nameN">value_atribute_N</atribute>
   </atributes>
   <langstring lang="query_lang">query_2</langstring>
 </question>
  ...
</questions>
```

Donde cada pregunta contiene un atributo único llamado *id* y una serie de atributos opcionales. Al final, y en la etiqueta *langstring* se introduciría la pregunta indicando en el atributo *lang* el idioma de la pregunta.

B.9.3. Búsqueda en modo cliente/servidor

Al igual que una búsqueda interactiva, a la hora de buscar una colección de preguntas se puede escoger tanto el modo cliente/servidor como

en modo local. El comando para iniciar una búsqueda a partir de un fichero de preguntas es:

```
jpm runs.xml RetrievalPassages [-client/server yes]
                [-question_format (clef|jirs)]
                -questions_file qfile -passages_file pfile
```

Donde:

client/server Al igual que en una búsqueda interactiva, este parámetro, que por defecto es *yes* indica que trabajará en modo cliente/servidor. Para ello es necesario haber lanzado inicialmente el servidor.

question_format El formato del archivo de preguntas. Por defecto es el del CLEF.

questions_file Un fichero con la lista de preguntas en algún formato explicado anteriormente.

passages_file El nombre del archivo XML que generará con los pasajes obtenidos. Este archivo tiene la siguiente sintaxis:

```
<questions>
 <question id="id_question">
  <langstring lang="qlang">question_1</langstring>
  <passages>
   <passage id="idpassage" lang="plang" doc="doc_id"
    sim="psimilarity">text_passage</passage>
   <passage id="idpassage" lang="plang" doc="doc_id"
    sim="psimilarity">text_passage</passage>
  </passages>
 </question>
 <question id="id_question">
  <langstring lang="qlang">question_2</langstring>
  <passages>
   <passage id="idpassage" lang="plang" doc="doc_id"
    sim="psimilarity">text_passage</passage>
   <passage id="idpassage" lang="plang" doc="doc_id"
    sim="1.0">text_passage</passage>
  </passages>
 </question>
  ...
</questions>
```

Donde *id_question* y *qlang* es el identificador y el idioma de la pregunta; *idpassage*, *plang*, *doc_id* y *psimilarity* son el identificador

del pasaje, el idioma, el identificador del documento al que pertenece el pasaje y la similitud que le da el sistema a ese pasaje respectivamente.

B.9.4. Búsqueda en modo local

El principal problema del modelo cliente/servidor es que hay que desconectar y relanzar el servidor cuando se requiere cambiar algún parámetro. Muchas veces se querrá realizar un barrido de parámetros de forma automática para encontrar aquellos parámetros que mejoren los resultados. Para ello se puede lanzar la búsqueda de una colección de preguntas de modo local, así, cada ejecución puede implicar un cambio parámetros. Esto se consigue con el siguiente comando:

```
jpm runs.xml RetrievalPassages -client/server no
          [-npassages num] [-question_format (clef|jirs)]
              -questions_file qfile -passages_file pfile
            [-model (distance|termweight|simple|rw|svm)]
   [-filter (yes|no)] [-naddlines num] [-disfactor factor]
        [-reformulation (yes|no)] [-searchengine (rw|svm)]
             [-term_weither (soft-idf|ne-soft-idf|tf-idf)]
   [-nse_passages num] [-combination (union|intersection)]
```

Los parámetros para esta ejecución se describen en los apartados B.6 y B.9.3

B.10. Crear una tabla de cobertura

Una vez creado un archivo XML con los pasajes devueltos a través de una colección de preguntas, es aconsejable evaluar el sistema para ver, en qué medida, los pasajes contienen efectivamente la respuesta o no. Para ello, a parte del archivo XML de pasajes generado en el punto anterior, es necesario un archivo con las posibles respuestas a cada pregunta realizada. El fichero de respuesta tiene el siguiente formato:

```
id_question_1 regular_pattern_1_1
id_question_1 regular_pattern_1_2
...
id_question_1 regular_pattern_1_n1
id_question_2 regular_pattern_2_1
id_question_2 regular_pattern_2_2
```

```
...
id_question_2 regular_pattern_2_n2
...
id_question_m regular_pattern_m_1
id_question_m regular_pattern_m_2
...
id_question_m regular_pattern_m_nm
```

Es decir, que para cada pregunta puede haber una serie de patrones regulares con las posibles respuestas. Por ejemplo estos patrones extraídos de las respuestas del CLEF 2005 para la tarea de español:

```
0001 Bayerische Motoren Werke
0001 fabricantes de automoviles
0002 Fuerzas Armadas Revolucionarias
0003 (presidente|lider) (de Sudafrica|sudafricano|del CNA
|del Congreso Nacional Africano)
0003 lider negro Nelson Mandela en Sudafrica
...
```

Donde el primer campo es el identificador de la pregunta y el segundo (separado por un espacio) el patrón regular que valida dicha pregunta.

Con el archivo de respuestas y el archivo XML de pasajes, podemos generar una tabla mediante el siguiente comando:

```
jpm runs.xml CreateCoverageTable -passages_file pfile
                  -correct_answers afile -table_file tfile
                                        [-num_passages num]
```

Donde:

passages_file Nombre del fichero XML que contiene los pasajes devueltos por el sistema para cada pregunta.

correct_answers El fichero que incluye los patrones regulares que validan la respuesta para cada pregunta como se especifica arriba.

table_file Fichero con la tabla de resultados de cobertura. Esta tabla tiene el siguiente formato:

```
          doc 1  2  3  4  5  6  7  8  9  10      total
query
0001        0  0  0  0  0  0  0  0  0  0        0
0002        0  0  0  0  1  0  0  0  1  1        3
```

```
0003    0  0  1  1  0  1  0  0  0  0    3
0004    0  0  1  0  0  0  0  1  1  0    3
0005    0  0  0  0  1  1  1  1  1  0    5
0006    1  1  1  1  1  1  1  1  1  1    10
0007    0  0  0  0  0  0  0  0  0  0    0
0008    1  1  1  1  1  1  0  0  0  0    6
0009    0  0  0  0  0  0  0  0  0  0    0
0010    1  0  0  0  1  0  0  0  0  0    2
0011    1  1  1  1  1  1  1  1  1  0    9
0012    0  0  0  0  0  0  0  0  0  0    0
0013    0  0  0  0  0  0  0  0  0  0    0
0014    0  0  0  0  0  0  0  0  0  0    0
0015    0  0  0  0  0  0  0  0  0  0    NIL
...
```

Donde la primera columna es el identificador de la consulta o pregunta y el resto de columnas es 1 o 0 dependiendo si en el pasaje i-ésimo se encuentra o no se encuentra la respuesta. La última columna es el número de respuestas encontrados en el pasaje. Cuando en esta columna aparezca NIL representa que, para esa pregunta, no había respuesta en la colección de documentos.

num_passages El número máximo de pasajes que revisará por pregunta, por defecto 20. Si la pregunta no tiene tantos pasajes lo rellena de ceros hasta completar el número especificado.

El formato de estas tablas de coberturas es muy sencillo de procesar. Es más, con JIRS se ajunta una serie de shell scripts que permiten obtener datos de estas tablas. Estos scripts (que se incluyen en la carpeta *scripts/*) son:

coverage Calcula la cobertura de una tabla de cobertura de JIRS para los n primeros pasajes. Su sintaxis es la siguiente:

```
coverage n < file
```

Donde n es el número de pasajes por pregunta a examinar y $file$ el archivo con la tabla de cobertura.

coveragebytype Obtiene la cobertura por tipo de pregunta. Su sintaxis es la siguiente:

```
coveragebytype coverage_table questions_file n
```

Donde *coverage_table* es la tabla de cobertura, *questions_file* el fichero de preguntas en formato CLEF y n el número de pasajes a examinar.

jirs2csv Devuelve las coberturas examinando primero el primer pasaje de cada pasaje, después los dos primeros, así hasta un número máximo de pasajes. Su sintaxis es:

```
jirs2csv file n
```

Donde $file$ es la tabla de cobertura y n es número de pasaje máximo a examinar por pregunta.

mrr Calcula el Media Reciprocal Rank de los pasajes obtenidos a partir de la tabla de cobertura hasta los n primeros pasajes. La sintaxis es:

```
mrr n < file
```

Donde n es el número de pasajes a examinar por pregunta y $file$ la tabla de cobertura.

B.11. Adaptar JIRS a nuevos idiomas no soportados

JIRS está configurado para trabajar con los idiomas de inglés, español, francés, italiano, alemán, árabe y urdu. En esta sección veremos cómo podemos adaptar JIRS para trabajar con nuevos idiomas que, inicialmente, no soporta.

A la hora de adaptar JIRS a un nuevo idioma es necesario aportar una serie de archivos de configuración que listo a continuación:

- Una lista de palabras comunes del nuevo idioma.

- Una lista de palabras interrogativas para el nuevo idioma.
- Una colección de documentos en formato del CLEF o TREC con texto en el idioma elegido.
- Un conjunto de preguntas.
- Un conjunto de respuestas.
- Reglas de reemplazo de caracteres especiales.
- Reglas de separación de caracteres de puntuación.

La lista de palabras comunes, los documentos y el conjunto de preguntas y respuestas deben estar en los formatos explicados en las secciones B.4, y B.9. El archivo de palabras comunes, como se comentaba en la sección B.4, es opcional si no se quiere eliminar dichas palabras durante la indexación. El listado de palabras interrogativas debe tener el siguiente formato:

```
<rules>
  <rule from="patrón_1" to=""/>
  <rule from="patrón_2" to=""/>
  ...
</rules>
```

Donde *patrón_ n* tendrá cada uno de los patrones de las preguntas a eliminar. El atributo *to* indica por qué será sustituido ese patrón, en este caso por nada. Por ejemplo:

```
<rules>
  <rule from="(Q|q)u(e|ien(es)?)␣" to="" />
  <rule from="(C|c)(omo|uando|ual(es)?|uant.(s)?␣" to="" />
  <rule from="(D|d)onde␣" to="" />
  <rule from="A␣que␣" to="" />
  <rule from="(Dar|Nombre)␣" to="" />
</rules>
```

Esto eliminará las palabras interrogativas de las preguntas en español.

El archivo que contiene las reglas de reemplazo de caracteres especiales sirven para sustituir palabras acentuadas por sus equivalentes sin acentos. Su formato es el siguiente:

```
<replaces>
  <replace letters="char_set_1" by="text_1"/>
  <replace letters="char_set_2" by="text_2"/>
  ...
</replaces>
```

Por ejemplo, el archivo para español sería:

```
<replaces>
  <replace letters="áàäâ" by="a"/>
  <replace letters="ÁÀÄÂ" by="A"/>
  <replace letters="éèëê" by="e"/>
  <replace letters="ÉÈËÊ" by="E"/>
  <replace letters="íìïî" by="i"/>
  <replace letters="ÍÌÏÎ" by="I"/>
  <replace letters="óòöô" by="o"/>
  <replace letters="ÓÒÖÔ" by="O"/>
  <replace letters="úùüû" by="u"/>
  <replace letters="ÚÙÜÛ" by="U"/>
</replaces>
```

Esto eliminará todas las ocurrencias de todas las vocales acentuadas por su equivalente sin acentuar.

Por último, se requeriría un archivo que contuviera unas reglas para separar los caracteres de puntuación del resto de palabras. Este archivo tendría el mismo formato que el archivo para eliminar las palabras interrogativas. Un ejemplo para el español sería:

```
<rules>
  <rule from="(&quot;|\(|\)|,|;|\.)" to="␣$1␣"/>
  <rule from="␣␣" to="␣"/>
</rules>
```

Donde todos los caracteres de puntuación serían reemplazados por un espacio, el mismo carácter de puntuación y otro espacio a continuación. La última regla se usa para eliminar del texto los espacios dobles que la primera regla pueda probocar.

Por último sólo queda modificar el archivo de configuración para que acepte éste nuevo idioma. Para ello se debe modificar el parámetro *language* para que contenga el nombre del nuevo idioma. Después se deben añadir las reglas de reemplazo y separación de puntuación en los parámetros condicionales *replace_character_file* y *clean_text_file* del

archivo de configuración como aparece en el ejemplo:

```
<switch param="language">
  <case value="english|spanish|french|italian|german">
    <parameter name="replace_character_file">
      $config_files$latin_repl_chars.xml
    </parameter>
    <parameter name="clean_text_file">
      $config_files$latin_clean_text.xml
    </parameter>
  </case>
  <case value="arabic|urdu">
    <parameter name="replace_character_file">
      $config_files$arabic_repl_chars.xml
    </parameter>
    <parameter name="clean_text_file">
      $config_files$arabic_clean_text.xml
    </parameter>
  </case>
</switch>
```

Aquí se podría crear otro *case* para añadir el nuevo idioma e indicarle así las nuevas rutas a estos archivos.

B.12. Errores comunes

DOC element is espected in line 1 Este error aparece cuando los ficheros a indexar no contienen el formato SGML esperado del CLEF o del TREC. Esto puede ser debido a que no se ha especificado el formato de descompresión de los ficheros (mirar apartado B.4).

java.lang.OutOfMemoryError: Java heap space El problema aparece cuando no hay suficiente memoria virtual para el programa. Para solucionarlo se debe modificar la memoria virtual del sistema operativo. Lo recomendable es 1GB de memoria virtual.

Apéndice C
GNU General Public License

Java Information Retrieval System (JIRS) is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

JIRS is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with jirs; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

C.1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free

Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

C.2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The 'Program', below, refers to any such program or work, and a 'work based on the Program' means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term 'modification'.) Each licensee is addressed as 'you'.

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with section b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under

any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and 'any later version', you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free

programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.