PROYECTO DE GRADO

MEGÚSEARCH

(META MOTOR DE BÚSQUEDA)

MARCOS CAMPAL – EDUARDO FRASCHINI

2004

TUTOR: JUAN JOSÉ PRADA

INSTITUTO DE COMPUTACIÓN
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA
MONTEVIDEO, URUGUAY

RESUMEN

Con el desarrollo de *Internet* en los últimos años, la Recuperación de Información ha llegado también a los usuarios inexpertos, avanzando más allá de la barrera de los expertos en sistemas de información. Esto trae aparejado consigo el problema de generar un sistema que pueda ser utilizado por esta clase de usuarios.

Los sistemas de Recuperación de Información en un entorno *web* (buscadores) presentan serias dificultades a los usuarios principiantes en el momento de encontrar la información deseada. Estos reciben las palabras clave generadas por el usuario y envían la consulta. En ningún momento interactúan con el usuario para intentar obtener el verdadero sentido de la consulta, ni siquiera en casos en que la ambigüedad de los términos utilizados lo requiera.

Este trabajo presenta una interfaz alternativa para interactuar con estos usuarios que facilita la generación de la consulta con la cual se logren obtener resultados satisfactorios en los buscadores comunes.

El método consiste en: recibir las palabras claves del usuario, retornar los conceptos relacionados de una ontología para que este elija el área de interés de su consulta. Más adelante desambiguar el sentido de los términos de la consulta inicial, dando a elegir otros términos relacionados con los sentidos elegidos. Por último generar la consulta para un buscador específico.

Esta propuesta se implementa utilizando un recurso lingüístico como *WordNet* y una ontología de categorías *web* como la propuesta por el *Open Directory Project*.

ÍNDICE

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 El problema

La búsqueda de información ha avanzado en gran medida en los últimos treinta años. En ese momento fue cuando se dejó de buscar libro por libro en las bibliotecas, y se comenzaron a utilizar grandes bases de datos bibliográficas con información específica sobre cada área de conocimiento. En ese primer avance, la interacción del sistema con el usuario final era realizada por un experto en sistemas de información. Este se encargaba de traducir cierta necesidad en una búsqueda (o una secuencia de ellas) que el sistema pudiera comprender. Si bien es más costoso tener un técnico al servicio del sistema, se tiene la ventaja que este sabe exactamente como realizar la interacción de forma de obtener resultados satisfactorios. Además, el usuario evitaba enfrentarse con un sistema con el cual no tuvo capacitación para utilizarlo, ni siquiera un contacto fluido para aprender a través de sus errores.

Con la aparición y expansión de *Internet*, se comenzaron a implementar sistemas de búsqueda que ayudaran a encontrar información en ese repositorio tan grande y heterogéneo. Estos sistemas de búsqueda (llamados "buscadores") utilizan diferentes técnicas para encontrar resultados relevantes y ordenar estos según un criterio determinado. A su vez, se enfrentan directamente con el usuario final de la aplicación, lo que lleva a estos a desarrollar una interfaz con la que puedan interactuar usuarios no expertos en los sistemas de recuperación de información. De esta manera, un usuario con algo de experiencia en la búsqueda de información, encuentra lo que desea sin grandes dificultades. El problema es que la expansión no finalizó en los informáticos y gente vinculada con las computadoras, sino que llegó a una gran cantidad de hogares. Esto hizo que alguien que apenas tenía un conocimiento básico de como interactuar con una computadora realizara búsquedas en estos sistemas. En ese momento, los buscadores se vieron en la necesidad de simplificar al máximo la interfase de búsqueda, proveyendo también de sugerencias para realizar estas búsquedas y pequeños tutoriales introductorios a la utilización del sistema.

Sin embargo sucede que, por lo general, los usuarios no están interesados en "perder" tiempo leyendo tutoriales para aprender como utilizar correctamente estos sistemas. Para esto se propone diseñar una interfaz de interacción con el sistema que sea sencilla de usar, y se encargue de generar una consulta que se corresponda con los deseos del usuario y ayude a este a encontrar la información requerida.

1.2 Motivación

Como se comentó en la sección anterior, por lo general los usuarios no desean pasar por una etapa de aprendizaje para utilizar correctamente el sistema, sino que quieren comenzar a utilizar este cuando lo necesitan. Algunos usuarios, los que tienen contacto más fluido con esta clase de sistemas, aprenden a utilizarlos sin tener idea de como funcionan internamente. De cualquier manera, la mayoría no llega a realizar consultas lo suficientemente específicas para encontrar lo requerido. Estos usuarios frecuentemente dicen "en *Internet* no se encuentra nada", pero el problema radica en como estos especifican sus necesidades.

Hay relevamientos que anuncian que en promedio el largo de una consulta realizada en un buscador *web* es de 1.5 palabras [Pink94], incluso muchas de estas consultas son realizadas con una única palabra.

Se puede inferir que, en la mayoría de los casos, la consulta no está correctamente formulada si se tiene en cuenta:

- la ambigüedad del lenguaje
- la existencia de siglas iguales con distinto significado
- la existencia de nombres de empresas con palabras que tienen su propio sentido en el diccionario
- la diversidad del vocabulario utilizado en la red
- los diferentes niveles de conocimiento de los que publican información en la *web* sobre el área o temática a la que refieren.
- los diferentes niveles de conocimiento sobre un tema específico de los que buscan información en ese tópico.

Esto ayuda a explicar el porque sobre la dificultad de los usuarios para especificar claramente sus intereses. Para alivianar este problema, es necesario brindarle alguna clase de ayuda. Este es, entonces, el problema que motiva el siguiente trabajo.

1.3 Organización del documento

El resto del documento se organiza de la siguiente manera: en el Capítulo 2 se presentan algunas definiciones básicas que servirán como guía para continuar la lectura del trabajo.

A continuación (Capítulo 3) se expone un extracto del documento: "Estado del Arte en Recuperación de Información".

En el Capítulo 4 se presenta la motivación, los objetivos, la evolución y el desarrollo del trabajo en si mismo, junto con algunos ejemplos de su funcionamiento. Además se muestra el diseño especificado, las tecnologías utilizadas y una explicación de la arquitectura cliente/servidor propuesta. Más adelante se presentan algunos detalles de implementación (más profundamente desarrollados en los apéndices correspondientes), la definición de la base de datos y su correspondiente módulo de importación. Se especifica también la metodología de implantación del producto y las pautas a seguir para su utilización en gran escala. Por último, se muestran los mecanismos de experimentación llevados a cabo para probar el sistema.

El Capítulo 5 contiene las conclusiones junto con la evaluación del sistema y los resultados obtenidos.

En el Capítulo 6 se explican las direcciones a seguir para continuar el desarrollo del producto así como los temas a estudiar para extender el trabajo. Se exponen: extensiones al servidor, extensiones al cliente de la aplicación y otra clase de trabajos no directamente relacionados con la aplicación pero si con las metodologías utilizadas. Más específicamente, se proponen trabajos en el área de desambiguación del sentido de las palabras como detectar significados de términos y realizar traducciones de consultas y textos.

A continuación está la sección de glosario. En esta se encuentran las definiciones de los conceptos más importantes relacionados con el documento.

En la Sección Referencias se pueden encontrar citas a los documentos de los cuales se extrajo información para el desarrollo de este trabajo.

El Apéndice A contiene un desarrollo profundo de la implementación de los algoritmos más interesantes de la aplicación. Entre ellos: la realización de consultas en el directorio de categorías, la obtención de sugerencias para un término especificado, y la generación de consultas luego de obtenida toda la información relevante. También se especifica el mecanismo seguido para generar perfiles de usuario y hacer estos persistentes.

El Apéndice B especifica las principales funcionalidades relacionadas con la interfaz gráfica de la aplicación cliente.

El Apéndice C explica como se configura el servidor y cuales son los parámetros modificables sobre los algoritmos implementados.

En el Apéndice D se muestra el funcionamiento del programa importador de categorías de una ontología. También se especifica como debe utilizarse para que el servidor pueda interactuar correctamente con la base de datos generada.

Por último, el Apéndice E está dedicado a explicar como se guarda la información requerida por la aplicación cliente.

De aquí en adelante, se supone de vital importancia para la comprensión de este documento la lectura del Capítulo 2 para quienes no se encuentren relacionados con el área *Recuperación de Información*. Aquí se brindan las definiciones básicas que se utilizan luego en el documento. La parte central del trabajo (de lectura inevitable) está compuesta por el Capítulo 4 donde se detalla este y el Capítulo 5 donde se muestran resultados obtenidos y conclusiones finales.

CAPÍTULO 2

2 DEFINICIONES BÁSICAS

En el siguiente capítulo se explican algunos conceptos utilizados a lo largo del proyecto. La lectura de este capítulo es de especial importancia para la gente que no está vinculada al área de investigación "*Recuperación de Información*".

2.1 Recuperación de información

La sección está dedicada a brindar una visión sobre los conceptos relacionados con la obtención de información a partir de un conjunto de documentos.

2.1.1 Introducción

El concepto *Recuperación de Información* (*IR*) suele definirse como la manera de obtener información relevante de un repositorio de documentos en respuesta a la consulta de un usuario. Esta consulta es especificada en lenguaje natural a través de un grupo de palabras claves.

Según Baeza [Baez99], la *Recuperación de Información* es la representación, almacenamiento, organización y acceso a ítems de información. Extendiendo los conceptos de esta definición:

- Representación: establece la información que se ve a guardar del documento.
- Almacenamiento: determina a través de que estructuras de datos se van a guardar los documentos.
- Organización: define jerarquías entre los documentos, temáticas, etc.
- Acceso a la información: determina la manera en que esta información va a ser accesible.

Esta representación de los documentos (o ítems) no es un problema fácil de resolver, tampoco su organización para poder otorgar resultados relevantes ante una consulta específica.

En *IR* resulta también interesante conocer, en determinados casos, datos que sean parcialmente relevantes para luego elegir los que mejor se ajustan a la consulta realizada. Esto debería ser tenido en cuenta al diseñar los puntos antes descritos.

Un *Sistema de Recuperación de Información* no informa al usuario sobre el tema que éste desea, sino que únicamente retorna una respuesta sobre la existencia, en el repositorio, de documentos relacionados con su búsqueda. El proceso de obtención de documentos relevantes a la consulta planteada no resulta nada simple debido a la ambigüedad del lenguaje y la complejidad semántica del vocabulario entre otras cosas.

Ejemplos de *Sistemas de Recuperación de Información* son los catálogos de las bibliotecas, donde cada entrada del catálogo es ejemplar de un documento ofrecido. En general, los documentos están identificados por un conjunto de palabras claves entonces el usuario puede especificar sus áreas de interés para obtener los documentos que se vinculan con sus deseos.

En definitiva, el propósito de esta clase de sistemas es retornar la mayor cantidad posible de documentos relevantes a una consulta de manera de satisfacer los deseos del usuario, obteniendo la menor cantidad de documentos no relevantes. En este caso, *relevancia* se define como la coherencia de los resultados recuperados respecto a las necesidades de adquisición de información del usuario.

Algunos de los problemas presentados al intentar recuperar documentos relevantes en un repositorio son los siguientes:

- Comprender el significado de los términos utilizados en la consulta. Resulta común que los términos ingresados tengan más de un sentido.
- Enfrentarse a la diversidad del lenguaje para especificar los mismos conceptos. Los documentos están escritos por personas diferentes que pueden haber utilizados diferentes términos para los mismos conceptos. Resulta entonces complicado obtener los documentos relevantes para un conjunto dado de términos.

A partir de las definiciones vistas, el término no incluye el área *Respuesta a Preguntas*, relacionado con sistemas que se encargan de procesar una pregunta realizada generalmente en lenguaje natural, para luego generar una respuesta con información relevante para el usuario. Tampoco incluye el tema *Recuperación de Datos* que se relaciona más específicamente con la obtención de datos, no únicamente relevantes sino aquellos que respondan directamente a la consulta (no alcanza con retornar lo que "mejor se ajusta" a la consulta). Generalmente, se realiza una especificación formal de este tipo de consultas.

Conceptos relacionados:

Dentro del área, se pueden encontrar también los siguientes conceptos:

- **Análisis automático de texto**: incluye las maneras de obtener un resumen del documento y representarlo adecuadamente en el repositorio. (este punto no será tratado en el documento).
- **Clasificación automática**: son los métodos de agrupación (*clustering*) de la información.
- **Estrategias de búsqueda**: para encontrar la información deseada.
- **Evaluación del sistema**: incluye las maneras en que estos sistemas son evaluados.

2.1.2 Recuperación de información en Internet

Con el desarrollo de *Internet* el concepto de *Recuperación de Información* se ha comenzado a utilizar para representar la búsqueda de páginas *web*, vistas ahora como los documentos en *IR* clásico.

Acompañado al progreso y difusión de la red de redes, se presentó la creación de los motores de búsqueda (comúnmente denominados "buscadores") encargados de obtener documentos relevantes frente a una consulta planteada mediante diferentes metodologías.

Esto trajo también nuevas complicaciones al enfrentarse al problema de recuperar información de *Internet*:

- Estructuración de los documentos: no existe una estructura uniforme para los documentos. Aquí no se presenta la necesidad de cumplir con ningún estándar ni control de calidad, cada uno publica lo que desea.
- Heterogeneidad de los datos: como se comentó en el ítem anterior, al no haber estándares a seguir, la información puede estar representada en distintos formatos, e incluso medios de comunicación, ya sean como: imágenes, texto o video, agregando además, la diversidad idiomática.
- Redundancia de la información: de alguna manera, el concepto de *Derechos de Autor* no es muy tenido en cuenta con el surgimiento de la red. La gente copia la información que le resulta interesante y la publica en su página personal, citando o no a quien la haya escrito inicialmente.
- Calidad de los datos: al no existir un proceso de control de lo publicado, suele suceder que se encuentren documentos pobremente escritos o incluso con información incorrecta.
- Volatilidad de los documentos: no existen restricciones ni información sobre la duración de un documento en la red. Estos pueden desaparecer de un día para el otro, o ser modificados sin previo aviso.
- Volumen de información: el crecimiento de la red se está dando en forma exponencial, lo que hace realmente dificultoso mantener guardada e indizada toda esta información.

A continuación se detallan algunas propuestas para manejar algunos de los problemas mostrados.

- Estructuración de los documentos y heterogeneidad de los datos: se pueden traducir los documentos a un estándar de intercambio para que el usuario obtenga la información más fácilmente.
- Redundancia de la información: al momento de presentar los resultados relevantes, se filtran aquellos que estén repetidos.
- Calidad de los datos: se pueden verificar algunas fuentes de la cual provienen los datos. Por ejemplo, no resulta demasiado complicado obtener cuáles páginas utilizan el documento en cuestión como referencia.
- Volatilidad de los documentos: para esto hay que contar con sistemas que recorran la red en períodos cortos de tiempo.
- Volumen de información: aquí se debe contar con grandes sistemas de información capaces de soportar estos volúmenes de datos.

En lo que refiere al usuario, al problema de especificar correctamente sus deseos para obtener resultados relacionados se le suma ahora el de enfrentarse con el inmenso volumen de documentos que puede obtener como resultado. Para esto los buscadores realizan ordenamientos según diferentes índices de relevancia (ver Documento: "Estado del Arte en *Recuperación de Información*").

2.1.3 Metabuscadores

Estos son servicios disponibles en la *web* que sirven para utilizar varios buscadores (o *Sistemas de Recuperación de Información*) utilizando una única interfase de comunicación. El usuario ingresa la consulta en un formulario, y el *metabuscador* se encarga de:

- enviarla a los buscadores ofrecidos.
- obtener los resultados.
- reordenarlos según un criterio determinado por el usuario o por el sistema en si mismo.
- mostrarlos luego de estandarizar la tipografía.

Sus principales ventajas son:

- ofrecer una única interfaz de búsqueda.
- acelerar el proceso de búsqueda manual en varios buscadores.
- traducir la búsqueda a la especificación dada por la interfaz de cada buscador utilizado.
- aumentar el cubrimiento del espacio de documentos para realizar la búsqueda.
- mostrar los resultados en un formato adecuado.

Y sus principales desventajas:

- se pierde la capacidad de especificar la búsqueda para cada motor.
- no siempre los ordenamientos son adecuados, por tanto se pueden perder resultados que aparecieron como muy relevantes para un motor determinado.

No todos los *metabuscadores* son iguales, estos se diferencian principalmente en:

- la interfaz que proveen para especificar la consulta.
- los motores de búsqueda que ofrecen.
- la importancia que se le da a cada buscador.
- la manera en que ordenan los resultados.

Los *metabuscadores* suelen facilitar la búsqueda de información especializada, ya que pueden consultar sitios específicos del área en cuestión más allá de los motores estándar. Por ejemplo: si el *metabuscador* detecta

que se ingresó una consulta sobre información científica, podría buscar en *Research Index*[1] comúnmente conocido como *CiteSeer* donde se pueden encontrar una gran cantidad de publicaciones de investigación.

2.1.4 Medidores

De alguna manera hay que medir el desempeño de esta clase de sistemas. Para esto se han propuesto diferentes indicadores. Los más importantes pueden ser extendidos a modelos de *Recuperación de Información* en *Internet*. Estos son:

Precisión:

Se define como la cantidad de documentos relevantes sobre el total de resultados obtenidos.

$$\text{Precisión} = \frac{\text{Cantidad de documentos relevantes recuperados}}{\text{Cantidad de documentos recuperados}}$$

Este indicador mide que tan ajustados son los resultados obtenidos con respecto a la información requerida por el usuario que generó la consulta.

***Recall* (Sensibilidad):**

Es la cantidad de documentos relevantes recuperados sobre el total de documentos relevantes.

$$\text{Recall} = \frac{\text{Cantidad de documentos relevantes recuperados}}{\text{Cantidad total de documentos relevantes}}$$

Este mide cuan bien el buscador (o *metabuscador*) cubre el espacio de documentos relevantes.

A continuación se muestra un diagrama que especifica los conceptos utilizados para definir estos medidores, y ayuda a visualizar sus significados.

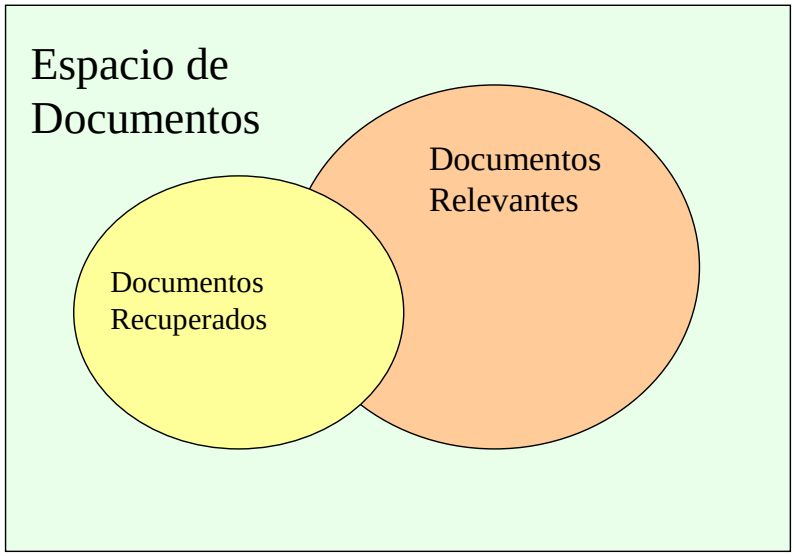

Figura 2.1 Diagrama de conjuntos de documentos

[1] Biblioteca digital de literatura científica: http://citeseer.ist.psu.edu/cs

Por lo general, sucede que estos indicadores varían inversamente. Es decir, cuando se modifica el sistema para que uno se incremente, el otro disminuye, y viceversa.

Los *Sistemas de Recuperación de Información* deben lidiar con la decisión de a que niveles mantener estos indicadores, ya que, cuando el usuario es experto en la temática sobre la cual busca información, le resulta fácil discriminar cuando un documento es relevante. Esto hace que sea más importante el *Recall* que la *Precisión*. En cambio, si el usuario es inexperto, sucede que pierde demasiado tiempo intentando determinar si un documento es relevante o no con respecto a la información que deseaba encontrar. Por tanto, es más importante la *Precisión* en este caso.

Un buen sistema de esta clase debe intentar maximizar la cantidad de documentos relevantes recuperados y minimizar la cantidad de documentos irrelevantes recuperados.

2.2 Recursos lingüísticos

En esta sección se detallan los recursos lingüísticos que se posee para realizar diferentes tareas relacionadas con la semántica del lenguaje.

2.2.1 Diccionarios

Un diccionario es una obra de consulta de significados para términos o palabras. En este caso se utilizan para expandir los términos, utilizados en su sentido apropiado, con otras palabras relacionadas semánticamente. Entre ellas: sinónimos, merónimos, hiperónimos e hipónimos.

Sinonimia:

Es la relación que vincula términos con un mismo significado.

Por ejemplo:

Los términos *obtener* y *adquirir* son sinónimos para el sentido: "tomar posesión de algo concreto o abstracto".

Hiperonimia:

Es una relación de subordinación entre términos. Se dice que un término es hiperónimo de otro si el último es una clase del primero.

Por ejemplo:

El término *construcción* es hiperónimo de *casa*. Y *casa* es hiperónimo de *mansión*.

Jerarquizando los términos de manera que los más generales están más arriba en la jerarquía y los más específicos más abajo, se puede decir que para dos términos donde uno está por encima del otro en la jerarquía, y estos están conectados por una relación, entonces el de arriba es hiperónimo del de abajo.

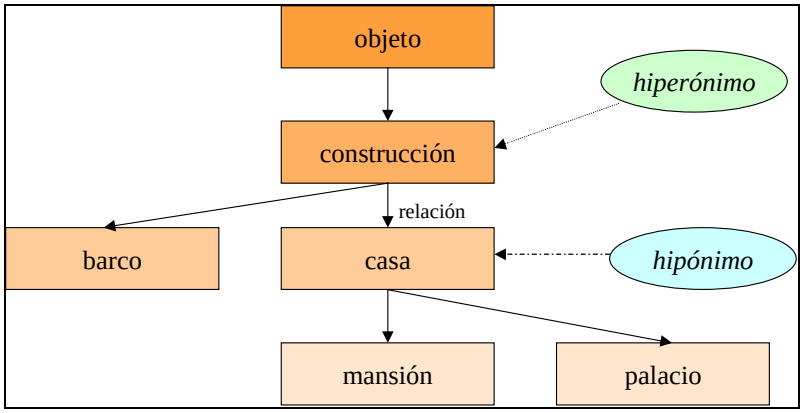

Figura 2.2 Relaciones jerárquicas

Hiponimia:

Es también una relación de subordinación entre términos. Más específicamente, es la relación opuesta a la hiperonimia. Se dice que un término es hipónimo de otro si el primero es una clase del último.

Por ejemplo:

Dado el sentido para la palabra *casa*: "Lugar que sirve como vivienda para una o más familias", se puede decir que los siguientes términos son algunos de sus hipónimos: bungalow, chalé, granja, hacienda, mansión y rancho.

Meronimia y Holonimia:

Meronimia es la relación semántica entre un término que define una parte y un término que define un todo.

Holonimia es la relación opuesta a la meronimia. Es el término que define el todo en la dependencia.

Por ejemplo:

- puerta (merónimo) – casa (holónimo): "La puerta es una parte de la casa".
- quijada (merónimo) – yegua (holónimo): "La quijada es la mandíbula de la yegua".
- brazo (merónimo) – cuerpo (holónimo): "Los brazos forman parte del cuerpo".
- sépalo (merónimo) – flor (holónimo): "El sépalo es parte de la flor".

Están sujetas a construcciones posesivas inalienables (que no se pueden transmitir).

Los siguientes no son ejemplos de relaciones de meronímia y holonimia:

- "La casa tiene un lavarropas".
- "El hombre tiene un perro".

Además, cabe notar que son diferentes de la hiponimia e hipernimia: un brazo no es una clase de cuerpo.

Homonimia:

Es una cualidad que se le da a un término cuando tiene la facultad de representar diferentes conceptos.

Por ejemplo:

Banco, que representa:

- Un lugar para sentarse.

- Una entidad financiera.

Polisemia:

Es la propiedad de los términos de tener varios significados.

Por ejemplo:

Agotar, que significa:
- Extraer todo el líquido que hay en una capacidad cualquiera.
- Gastar del todo, consumir.

2.2.2 Tesauros

Un *Tesauro* es una clase de *Diccionario* que contiene términos del lenguaje natural y su equivalencia con los términos normalizados del lenguaje documental. También contiene relaciones de diferentes clases entre términos. Se caracterizan además por preponderar las relaciones asociativas por sobre las jerarquías de conceptos. Esto no quiere decir que no haya relaciones jerárquicas, sino que se centra en las relaciones de tipo asociativo.

Resultan útiles para facilitar el procesamiento de textos a nivel informático, sugiriendo, entre otras cosas, alternativas para la búsqueda de información.

La estructura de un tesauro está basada en las relaciones entre conceptos. Estas relaciones pueden ser de tipo:
- Jerárquicas.
- De similitud.
- De preferencia.

Jerárquicas:

Indican términos más amplios o más específicos para cada concepto. Se pueden ejemplificar con las relaciones:
- mueble – mesa
- máquina – computadora
- animal – humano

Estas relaciones se indican por los indicadores:
- Término más amplio (*Broader Term - BT*)
- Término más específico (*Narrower Term - NT*)

De similitud:

Estas marcan términos que están relacionados conceptualmente, pero que no tienen orden jerárquico.

Por ejemplo:
- casa – hogar
- pared – muro

Estas suelen estar indicadas por el indicador:
- Término relacionado (*Related Term – RT*)

De preferencia:

Se utilizan para indicar el descriptor preferido dentro de un conjunto de sinónimo.

Hay dos clases de relaciones:

- Usar (*USE*) y su recíproco: Usado Por (*Used For – UF*)
- Ver (*SEE*) y su recíproco: Visto Por (*Seen For – SF*)

En los tesauros se utiliza el término *USE* para indicar el término preferido en caso de sinónimos. Por ejemplo, para: "*droga*" se usa "*sustancia dependiente*".

droga USE sustancia dependiente

En el caso de los homónimos se utiliza *SEE*. Por ejemplo, para *fabricación* se recomiendo observar *proceso*.

fabricación SEE proceso.

2.2.3 Ontologías

Existen diferentes acepciones para el término, siendo todos bastantes similares se pueden exponer algunos para dar una idea de lo que se quiere denotar.

Según Neches [Nech91], esta se define como un instrumento que define los términos básicos y relaciones a partir del vocabulario de un área así como las reglas de combinación de estos términos y relaciones para definir extensiones a un vocabulario.

A juzgar por Guerrero y Lozano [Guer99], las ontologías son construcciones que estructuran contenidos explícitos y que son capaces de codificar las reglas implícitas de una parte de la realidad.

De alguna manera, las ontologías ordenan conceptos y mostrando relaciones entre estos de forma de especificar temáticas o conceptos más generales, también llamados hiperónimos (más arriba en la jerarquía) y otros más específicos (más abajo), llamados hipónimos.

Esta conceptualización debe ser representable de manera formal y de ser posible para que una computadora la pueda utilizar. Las ontologías están organizadas en una taxonomía definida a partir de las relaciones entre los conceptos.

La imagen muestra la estructura principal de una ontología junto con algunas de las relaciones representadas:

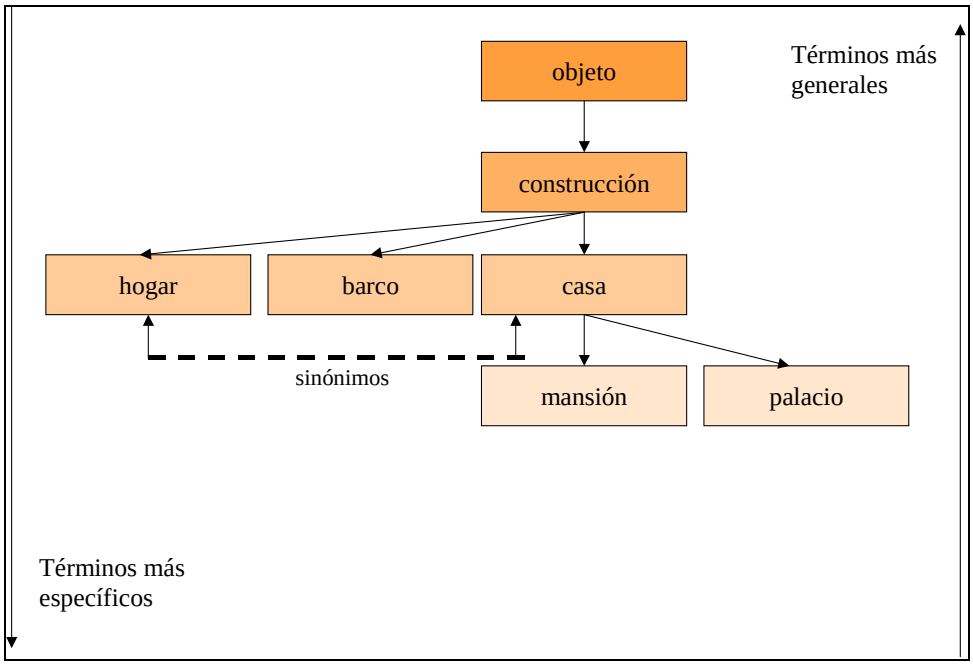

Figura 2.3 Estructura general de una ontología

En el documento, los términos que forman parte de toda ontología serán denominados categorías para no resultar confundidas con las palabras que son ingresadas por el usuario al sistema, ni con aquellas que son elegidas luego de la desambiguación de estos términos.

Cuentan con los siguientes componentes para representar el conocimiento:

- **Conceptos:** ideas que se trata de especificar.
- **Relaciones:** estas representan los enlaces entre conceptos del dominio. Especifican, de alguna manera, la taxonomía de este dominio.
- **Funciones:** son una clase específica de relación. En estas se identifican los elementos mediante el cálculo de un valor derivado de otros conceptos de la ontología.
- **Instancias:** estas se utilizan para representar un objeto determinado de un concepto.
- **Axiomas:** son especificaciones declaradas sobre las relaciones que deben cumplir los elementos de la ontología. Por ejemplo: "Si A es de clase B y B es subclase de C entonces A es de un tipo subclase de C". Estos permiten inferir conocimiento no especificado explícitamente en la ontología.

A partir de esta serie de conceptos se especifica la taxonomía que es la estructura principal que define a las ontologías.

2.2.4 Diferencias entre Tesauros y Ontologías

Varios autores se han centrado en esta área, tratando de especificar cuáles son las características que hacen diferentes estos dos recursos semánticos.

Según Qin y Palin [QinP00], las ontologías son superiores a los tesauros ya que:

- presentan un nivel de concepción y descripción del vocabulario.
- se caracterizan por un desarrollo semántico más profundo para las relaciones taxonómicas y las relaciones cruzadas.
- son más útiles en el momento de describir realidades. Esto es debido a la manera de definir relaciones entre los conceptos.

Por el mismo camino van Ding y Foo [Ding02] que consideran que las diferencias más importantes entre una ontología y un tesauro se sitúan en el nivel de abstracción y en la expresividad que puede llegar a proporcionar sobre los conceptos, sobre todo en el momento de ser utilizados por máquinas.

De esta manera comentan:

- Una ontología puede ajustarse a determinados requerimientos.
- Estas potencian la definición de conceptos a través de sus relaciones.
- Promueven la normalización y utilización del conocimiento.
- Agregan valor a los tesauros ya que representan los conceptos desde una semántica más profunda. Además, los muestran en una jerarquías conceptual que enriquece las relaciones entre conceptos.

2.2.5 Stemmers

Un *stemmer* es una herramienta que obtiene la forma de una palabra luego de que todos sus afijos (prefijos y sufijos) fueron removidos.

Por ejemplo:

Palabra	Raíz (*Stem*)
apple	appl
apples	appl
artistic	art
arts	art

Figura 2.4 *Stemming* de términos

2.2.6 Lematizadores

Un lematizador es un sistema que obtiene una palabra en su forma más simple. Esta es la que aparece como entrada en diccionarios y tesauros.

Por ejemplo:

Palabra	Lema
apple	apple
apples	apple
artistic	artistic
arts	art

Figura 2.5 Lematización de términos

2.2.7 Comparación: stemmers y lematizadores

Estas herramientas ofrecen la posibilidad de obtener un término (o parte de él) que represente las derivaciones de una palabra.

El objetivo de utilizar estos sistemas es: obtener la raíz de una palabra y no quedarse con morfemas flexivos o derivativos que no se utilizan en las ontologías manejadas.

Los algoritmos que se encuentran disponibles para realizar la tarea de *stemming* son bastantes simples y de bajo costo computacional. No es así para los lematizadores. Por tanto, resulta menos costoso utilizar *stemmers* para unificar palabras que tienen una misma raíz.

Se estudió la utilización del algoritmo de *Porter* [Port80] para este problema. Sin embargo, se encontraron algunas fallas.

Por ejemplo:

Palabra	**(*Porter Stem*)**
business	busi
busy	busi

Figura 2.6 *Stemming* utilizando el algoritmo de *Porter*

Aquí se puede observar que según el algoritmo de *Porter*, los términos "negocios" y "ocupado" tienen el mismo significado, o más específicamente, la misma raíz. Esto haría que el sistema que utilice este algoritmo retorne resultados relacionados con "negocios" cuando se ingrese el término "ocupado" en una consulta.

CAPÍTULO 3

3 MODELOS EN *IR*

En este capítulo se presenta un pequeño extracto del documento: "Estado del Arte en *Recuperación de Información*".

A continuación se detallan diferentes metodologías para implementar *Sistemas de Recuperación de Información*. Estos sistemas, como fue comentado en el Capítulo 2, se encargan de:

- guardar la información de los documentos del repositorio.
- indizar esta información para que sea fácil de obtener.
- implementar mecanismos de *Recuperación de Información* a través de una interfase de búsqueda para el usuario.

3.1 Modelos clásicos

Los modelos clásicos de *IR* fueron los primeros presentados cuando se comentó a atacar la problemática antes descrita.

Estos consideran que cada documento puede ser especificado como un conjunto de palabras representativas denominadas términos índice. Un término índice es simplemente una palabra (del documento) cuya semántica ayuda a recordar los temas principales tratados por el documento. Por lo tanto, los términos índice sirven para indexar y resumir los documentos que los contienen. De esta manera, se utilizan estos términos para obtener rápido acceso a la información del documento.

En general, los términos índice son sustantivos, ya que éstos generalmente tiene significado por si mismos. Los adjetivos, adverbios y conectivos son menos útiles como términos índice (aunque existen sistemas que los utilizan).

Es de suponer, que cada término índice de un documento tiene distinta relevancia. Por lo tanto, se pondera cada índice con algún valor que represente esta información. Estos pesos se consideran independientes entre si.

Por lo general, los modelos clásicos guardan los documentos como un conjunto de términos índice. Esto genera una gran pérdida de información. Sin embargo, simplifica en gran forma las técnicas de indexado y priorización de los documentos.

Los tres modelos clásicos más conocidos son:

- Modelo Booleano
- Modelo Vectorial
- Modelo Probabilista

3.1.1 Modelo Booleano

Los documentos son representados con vectores de pesos binarios que muestran la aparición (o no) de cada término.

Las consultas se expresan mediante predicados booleanos. Por ejemplo:

((k1 or k2) and k3) , k1, k2 y k3 términos.

Estas son traducidas a una expresión Booleana en su forma normal disyuntiva: donde las posiciones 1, 2 y 3 de cada vector representan a k1, k2 y k3 respectivamente. En este caso, quedaría (1,0,1) *or* (0,1,1).

Se dice que un documento es relevante a una consulta, si alguno de los componentes conjuntivos de la consulta concuerda con la expresión del documento.

Se puede ver entonces que el modelo puede predecir si un documento es relevante o no frente a una consulta, pero no puede determinar una concordancia parcial entre ellos. Además, tampoco es capaz de medir el grado de relevancia de un documento frente a la consulta, lo que impide ordenar los documentos recuperados por grado de relevancia.

3.1.2 Modelo Vectorial

El *Modelo Vectorial* intenta medir la concordancia parcial entre documentos y consultas. Con esta finalidad, se le asigna un peso no binario a los términos que aparecen en los documentos y las consultas. En general, estos pesos están dados por la frecuencia de cada término en el documento Al final, son utilizados para medir el grado de similitud entre cada uno de los documentos del repositorio y la consulta formulada por el usuario. La respuesta dada por el sistema es entonces una lista de documentos ordenada en forma decreciente respecto a la similitud con la consulta.

3.1.3 Modelo Probabilista

El *Modelo Probabilista* se basa en el siguiente principio: La probabilidad de que un usuario encuentre relevante a un documento depende únicamente de las representaciones de la consulta formulada y del documento.

Para un documento dado: se define grado de semejanza con una consulta como la probabilidad de que pertenezca al conjunto de documentos relevantes, sobre la probabilidad de que pertenezca a los no relevantes. Realizando una estimación inicial de estos valores, se obtiene un primer ordenamiento de los documentos que luego es mejorado a través de la interacción con el usuario o utilizando mecanismos automáticos basados en la distribución de los términos de la consulta en el conjunto de documentos recuperados.

3.2 Modelos avanzados

A partir de los modelos simples surgieron otros que resuelven algunos de los problemas que estos tienen. Sin embargo, estos son más difíciles de comprender y más costosos de implementar.

Los modelos comentados son:

- Modelo de Conjuntos difusos
- Modelo Booleano Extendido
- Modelo de Espacio de Vectores Generalizado

Estos modelos son comentados en el documento: "Estado del Arte en *Recuperación de Información*".

3.3 Nuevas técnicas

A continuación se detallan brevemente ciertas técnicas novedosas utilizadas en *Recuperación de Información*. Algunas de ellas son tenidas en cuenta en este trabajo.

3.3.1 Agrupación de Términos

Surge a partir de la dificultad de encontrar resultados más relevantes luego que un buscador retornó una lista de documentos. La propuesta es organizar los resultados en grupos (*clusters*) temáticos. De esta manera, el usuario selecciona el tema al cual refería su consulta para luego buscar los documentos que pertenecen a ese grupo.

3.3.2 Expansión de Consultas

Para atacar el mismo problema sobre el cual se propuso la técnica de agrupación de términos se presenta la metodología de expansión de consultas. Esta se basa en agregar términos relacionados con una consulta inicial de forma de aumentar la precisión de la misma.

Por lo general, estos sistemas ofrecen términos considerados relacionados al usuario, y este elige que términos desea agregar. Sin embargo, existen métodos que agregan automáticamente términos a la consulta según algún criterio particular.

Estos sistemas obtienen palabras de diferentes fuentes, entre ellas:

- consultas anteriores de usuarios
- interacción con el usuario
- tesauros
- ontologías
- retroalimentación a partir de los documentos retornados por una búsqueda

3.3.3 Perfiles de Usuario

Es un enfoque que se ha comenzando a estudiar recientemente. El concepto se basa en utilizar información característica del usuario para mejorar el resultado de una consulta.

Este método consta de dos etapas:

- modelado de usuarios.
- utilización del modelo.

Modelado de Usuarios:

Esto se puede realizar "observando" las actividades del usuario dentro de un período de tiempo. Se puede registrar información sobre:

- consultas que realiza
- documentos que selecciona
- documentos que recibe
- documentos que lee
- programas que utiliza

Se considera que este método invade las libertades y la privacidad del usuario.

Existe otro método más simple que consiste en darle un conjunto de términos al usuario y dejarlo elegir los que más le interesan, o simplemente dejarle especificar las áreas o temáticas que le interesan mediante palabras claves o descriptivas.

Utilización del modelo:

Existe una gran cantidad de métodos para utilizar esta clase de modelos. Entre ellos:

- Expandiendo la consulta: se agregan palabras relacionadas con los términos ingresados. Estas palabras son algunas de las que se encuentran en el perfil definido anteriormente.
- Filtrando resultados: se eliminan los resultados que no se consideran relacionados con el perfil especificado.

CAPÍTULO 4

4 META BUSCADOR

4.1 Introducción

Comúnmente sucede que las consultas realizadas por los usuarios resultan demasiado ambiguas y retornan gran cantidad de documentos no relacionados con lo requerido. Para filtrar estos documentos, se intenta precisar la consulta mediante diferentes métodos.

En este caso el sistema se basa en ofrecer al usuario, términos relacionados con los que ingresa. Esta metodología es denominada "Expansión de consultas" y tiene la interesante característica de servir, no solamente para reducir la cantidad de documentos retornados no relacionados, sino también para devolver más resultados relevantes.

Por ejemplo, considerando que el buscador acepta los operadores *or* y *and* (todos los buscadores conocidos lo hacen):

El usuario está intentando conseguir un trabajo e ingresa al sistema la palabra "*work*". Un sinónimo para el concepto que desea introducir el usuario sería "*employment*". A partir de esto, surgen dos posibles expansiones a la consulta:

- "*work or employment*". Esta retornaría más resultados relevantes que la consulta inicial.
- "*work and employment*" Aquí se puede observar que se obtienen menos resultados irrelevantes, o sea que la precisión de la respuesta aumentaría.

En este breve ejemplo se puede vislumbrar que la estrategia es muy amplia y permite ajustarse a diferentes necesidades.

Dentro de la metodología existe una división central basada en el hecho de si se utiliza al usuario para elegir términos a agregar en la expansión. Es de público conocimiento que el usuario quiere perder la menor cantidad de tiempo posible en la búsqueda de información. Sin embargo, los sistemas que devuelven una lista de posibles términos a utilizar obtienen mejores resultados que los que no lo hacen. Estos últimos, infieren los términos a utilizar a partir de datos como por ejemplo:

- Información de un perfil de usuario.
- Datos estadísticos.
- Comportamiento anterior.

En el sistema desarrollado en este trabajo se aplican las dos técnicas: una inicial en que se pide al usuario que elija conceptos relacionados, y una final en que el programa agrega otros términos automáticamente en caso de considerarlo oportuno.

4.2 Objetivos

Existe una gran cantidad de metas a superar en este trabajo. Sin embargo, estas pueden ser ordenadas con respecto a su importancia. A continuación se presenta una lista de estos objetivos ordenados comenzando por el más relevante.

4.2.1 Sistema de generación de consultas

El objetivo principal es desarrollar un sistema que ayude al usuario a generar una consulta de la manera más específica posible antes de ingresarla en un buscador.

Esto implica:

- Obtener el significado que el usuario desea.
- Agregar términos que hagan más precisa la consulta.
- Agregar sinónimos que ayuden a cubrir el área en forma más amplia.

Obtener el significado que el usuario desea:

El problema es más grande de lo que parece. La ambigüedad de los idiomas hace que una consulta de una o dos palabras sea ambigua en la mayoría de los casos. Por lo tanto, hay que obtener información extra para obtener el sentido apropiado para la consulta y poder utilizar esto en favor del usuario.

Agregar términos que hagan más precisa la consulta:

Una vez obtenido el significado apropiado (en caso de ser posible), se pasa a la siguiente etapa que es intentar adquirir nuevos términos, tal vez más específicos, para mejorar la precisión de los resultados. Esto se consigue a través de hiperónimos, hipónimos y otras palabras relacionadas con los términos desambiguados.

Agregar sinónimos que ayuden a cubrir el área en forma más amplia:

Luego de conseguir el nivel de precisión deseado para la consulta, se trata de obtener la mayor cantidad de documentos relacionados.

Debido a la diversidad del lenguaje, sucede que en algunos casos el documento está relacionado pero no utiliza los términos que el usuario ingresó (o el sistema agregó en la etapa anterior). Como los buscadores se encargan (en su mayoría) de encontrar apariciones de términos en documentos, resulta que este tipo de documentos no aparece como relevante. Para solucionar este problema, se utilizan sinónimos de las palabras ingresadas, de manera de obtener también estos documentos.

Estas no son tareas sencillas y hay cientos de documentos que las tratan sin obtener resultados demasiado satisfactorios. Igualmente, para intentar acercarse a una solución aceptable al problema, se evaluaron distintas metodologías que se exponen más adelante.

4.2.2 Interfaz simple y amigable

Resulta importante que la interfaz sea intuitiva y haga el sistema fácil de manejar. Cabe recordar que este es un sistema de ayuda a usuarios inexpertos, por lo tanto, la interfaz no debe estar desarrollada para profesionales en el área de *Recuperación de Información*.

Esto implica un sinnúmero de detalles. Entre ellos:

- Botones con dibujos explicativos de las funciones que cumplen.
- Mensajes que informen de las funciones de los botones.
- No excederse en la cantidad de funcionalidades que ofrece el sistema. De otra forma, el usuario podría verse abrumado y perderse las opciones principales.
- Desarrollo de un pequeño documento de ayuda que estimule una utilización rápida y eficiente del sistema.
- Extensión del idioma de la interfaz para que el usuario se sienta cómodo al utilizarla.

4.2.3 Utilizar una ontología que categorice la información

Es deseado utilizar una ontología que ordene conceptos y categorice de alguna manera la información de la red.

De esta manera, ubicando los términos ingresados por el usuario en la ontología, es posible a través de los conceptos relacionados en la taxonomía obtener el verdadero sentido de cada término.

Una vez ubicados los términos, se puede utilizar la información dada por las relaciones de la jerarquía para obtener términos más específicos y otros relacionados con cada palabra.

4.2.4 División de tareas entre cliente y servidor

No es de esperar que el cliente que va a utilizar la aplicación tenga los conocimientos necesarios para poder instalar: un diccionario, una base de datos e importar los datos que esta requiera. Por tanto la aplicación deberá diseñarse con una arquitectura cliente/servidor que permita al usuario olvidarse de esos detalles.

Esta división aportará además, la posibilidad de actualizar el directorio de categorías sin que el usuario deba hacer algún cambio en su aplicación. También se podrán hacer modificaciones en los parámetros del sistema para regular que tan audaz pueda ser este en el momento de desambiguar palabras o agregar términos a la consulta.

4.2.5 Algoritmos eficientes y con tiempos de respuesta cortos

Las restricciones de tiempo son un problema crucial en el correcto desempeño del sistema y en el éxito que este tenga. Hay que tener en cuenta que los usuarios que utilizan buscadores están acostumbrados a tiempos de respuesta breves. Por tanto, los algoritmos a desarrollar deben tomar en cuenta ciertos límites de tiempo aceptables para el usuario.

Dada la gran cantidad de datos que se maneja, es esencial que los problemas se solucionen de manera eficiente para obtener los resultados deseados. Además, al estar desarrollando una aplicación cliente/servidor, entra en juego el tema de la transferencia de datos entre el cliente y el servidor. Por tanto, hay que minimizar estos pasajes para alcanzar límites de tiempo aceptables.

4.2.6 Estudio de nuevas metodologías

Es también objetivo del proyecto, investigar nuevos métodos para intentar resolver los problemas planteados.

En el área de desambiguación del sentido de las palabras, hay gran cantidad de trabajos y son pocos los que obtienen resultados satisfactorios.

En lo que refiere a expansión de consultas existen nuevos trabajos interesantes. Además, hay gran expectativa en encontrar sistemas que mejoren la precisión y el grado de cubrimiento del espacio de búsqueda.

Para obtener más información sobre los trabajos y las metodologías relevadas dirigirse al documento: "Estado del Arte en *Recuperación de Información*".

4.2.7 Sistema flexible a cambio e inserción de nuevas ontologías

Si bien, la idea inicial es trabajar con la ontología ofrecida por el *Open Directory Project* (*ODP*), el sistema a desarrollar debe ser lo suficientemente flexible para poder soportar diferentes ontologías sobre distintas temáticas.

Este objetivo permitiría al sistema poder extenderse para ayudar en la búsqueda de información en terrenos especializados, donde es aun más difícil encontrar información. Incluso puede utilizarse fuera de *Internet*, en una red local, facilitando la búsqueda de información en bases de datos privadas.

4.2.8 Sistema flexible a cambio del diccionario

Al igual que en el ítem anterior, el sistema debe ser flexible a la posibilidad de cambiar el diccionario utilizado por uno de dominio especializado u otro con más información.

Esto permitiría la obtención de sinónimos para palabras técnicas que no se encuentran en diccionarios de propósito general.

4.2.9 Flexibilidad a cambios en el idioma de dominio de la aplicación

Este punto surge como consecuencia de los dos anteriores. Al poder modificar la ontología y el diccionario utilizados, el sistema debe poder funcionar correctamente en otro idioma diferente de para el cual fue pensado.

De esta manera, obteniendo las fuentes de información necesaria puede utilizarse la aplicación para generar consultas en el idioma que se desee. Otra aplicación que surge de esta funcionalidad es la posibilidad de traducir una consulta. Si las aplicaciones utilizadas como ayuda están disponibles en varios idiomas se puede hacer una traducción de la consulta siguiendo los pasos que llevaron a su creación para un idioma determinado.

4.3 Evolución del Sistema

Aquí se describe como fue variando la metodología diseñada hasta llegar a lo especificado en el ítem 4.4 (Descripción del Sistema). Como se podrá observar, el resultado final es alcanzado mediante un proceso de evaluación de diferentes técnicas y herramientas.

4.3.1 Expansión sugiriendo sinónimos de tesauros

La primera opción manejada fue realizar una expansión de términos basada en ofrecer al usuario una lista de definiciones de las palabras de la consulta, para que este pueda elegir el significado que más se ajusta a su deseo. Posteriormente se le ofrecerían términos relacionados en base a los sentidos seleccionados (uno para cada término ingresado).

Esta metodología es bastante simple de implementar utilizando un diccionario como *WordNet*. Sin embargo, es bastante tedioso para el usuario elegir un sentido para cada palabra, para luego elegir los términos con los cuales desea expandir la consulta. Por nombrar un ejemplo, la palabra "*work*" en *WordNet* consta de 34 significados diferentes, por lo que se tardaría un tiempo considerable en leer todas las definiciones y encontrar la apropiada.

Además, el enfoque no contempla algunos casos importantes. Estos casos surgen de los nombres de algunas compañías como *Apple Macintosh* o *Jaguar*[2], que son fruto de numerosas búsquedas en la red, y sus nombres no están definidos en el diccionario.

4.3.2 Expansión utilizando un perfil de usuario

Como se vio en el ítem anterior, resulta tedioso tener que elegir entre una gran cantidad de definiciones, el significado deseado para una palabra determinada. Con este método se intenta evitar al usuario dicho trabajo.

Previamente a la utilización del sistema, se debe definir un perfil de usuario a partir de ciertas categorías predefinidas, un conjunto de categorías posibles sería:

- *Art*

[2] Compañía inglesa productora de vehículos. Su sitio oficial en *Internet* es: http://www.jaguar.com/

- *Shopping*
- *Science*
- *Games*
- *Business*
- *Health*
- *Sports*
- *Society*
- *Animals*

Se propone al usuario elegir un subconjunto de categorías para luego utilizar el sistema.

Cuando se ingresa una consulta. El sistema elige entre todas las definiciones de cada palabra, la que tiene un mayor *Valor de Concordancia* con respecto al perfil seleccionado.

La noción de *Valor de Concordancia* es introducida de la siguiente manera:

- Se obtienen todos los hipónimos (términos más específicos) de los sentidos de las palabras seleccionadas en el perfil.
- Se genera un conjunto de palabras que contenga las raíces de los hipónimos de cada palabra del perfil. Este se llamará "*Conjunto perfil*".
- Se generan conjuntos de palabras a partir de las raíces de los sinónimos de cada definición encontrada en el diccionario. Se llamarán "*Conjunto Definición #*" donde # es el número de definición que genera el conjunto.
- Se buscan ocurrencias conjuntas entre el *Conjunto Perfil* con cada uno de los *Conjuntos Definición*. Cada ocurrencia de una raíz de un *Conjunto Definición* en el *Conjunto Perfil* suma 1 al valor de concordancia entre la definición y el perfil elegido.
- En caso de que no haya coincidencias con ningún *Conjunto Definición*, se obtienen los hiperónimos (términos más específicos) de cada definición, y sus raíces se agregan a cada conjunto para realizar las comparaciones nuevamente.

La definición elegida es la que ofrece candidatos para la expansión de la consulta. Estos candidatos pueden ser: sinónimos, hipónimos, hiperónimos, términos relacionados, etc.

EJEMPLOS

1. El usuario seleccionó la siguiente palabra para definir su perfil:

- *Science*

Se obtienen los sentidos y las categorías derivadas de los términos del perfil (hipónimos):

Science (2 sentidos)

1. science, scientific discipline -- (a particular branch of scientific knowledge; "the science of genetics")

   Lista de Hipónimos:

   - natural science -- (the sciences involved in the study of the physical world and its phenomena)
   - mathematics, math, maths -- (a science (or group of related sciences) dealing with the logic of quantity and shape and arrangement)
   - agronomy, scientific agriculture -- (the application of soil and plant sciences to land

management and crop production)
- agrobiology -- (the study of plant nutrition and growth especially as a way to increase crop yield)
- agrology -- (science of soils in relation to crops)
- architectonics, tectonics -- (the science of architecture)
- metallurgy -- (the science and technology of metals)
- metrology -- (the scientific study of measurement)
- nutrition -- (the scientific study of food and drink (especially in humans))
- psychology, psychological science -- (the science of mental life)
- information science, informatics, information processing, IP -- (the sciences concerned with gathering and manipulating and storing and retrieving and classifying recorded information)
- cognitive science -- (the field of science concerned with cognition; includes parts of cognitive psychology and linguistics and computer science and cognitive neuroscience and philosophy of mind)
- social science -- (the branch of science that studies society and the relationships of individual within a society)
- strategics -- (the science or art of strategy)
- systematics -- (the science of systematic classification)
- thanatology -- (the branch of science that studies death (especially its social and psychological aspects))
- cryptanalysis, cryptanalytics, cryptography, cryptology -- (the science of analyzing and deciphering codes and ciphers and cryptograms)
- linguistics -- (the scientific study of language)

2. skill, science -- (ability to produce solutions in some problem domain; "the skill of a well-trained boxer"; "the sweet science of pugilism")

   Lista de Hipónimos:

   - nose -- (a natural skill; "he has a nose for good deals")
   - virtuosity -- (technical skill or fluency or style exhibited by a virtuoso)

Figura 4.1 Sentidos otorgados por *WordNet* al término "*science*" junto con sus hipónimos

.
Se calcula el *Conjunto Perfil*:

{*agricultur, agrobiolog, agrolog, agronomi, architecton, cognit, cryptanalysi, cryptanalyt, cryptographi, cryptolog, disciplin, informat, inform, ip, linguist, math, mathemat, metallurgi, metrolog, natur, nose, nutrit, process, psycholog, psycholog, scienc, scientif, skill, social, strateg, systemat, tecton, thanatolog, virtuos*}

La consulta ingresada es: "*groups*" y los sentidos retornados por *WordNet* los siguientes:

Como sustantivo:

- group, grouping -- (any number of entities (members) considered as a unit)
- group, radical, chemical group -- ((chemistry) two or more atoms bound together as a single unit and forming part of a molecule)
- group, mathematical group -- (a set that is closed, associative, has an identity element and every element has an inverse)

Como verbo:

- group -- (arrange into a group or groups; "Can you group these shapes together?")
- group, aggroup -- (form a group or group together)

Figura 4.2 Sentidos otorgados por *WordNet* al término "*groups*"

Como se puede observar en la tercera definición para el sustantivo *groups*, está el sinónimo *mathematical group*, que va a coincidir con la raíz *mathemat* del *Conjunto Perfil*. Además, esta será la única raíz que coincidirá, lo que deja la definición número 3 como primera opción para elegir.

De esta manera, se ofrece al usuario el término *methamatical* para expandir su consulta.

Este enfoque tiene una gran ventaja que es evitar al usuario tener que leer todas las definiciones del diccionario para elegir la que más se ajusta a sus deseos. Sin embargo es común que no se pueda decidir cual significado elegir, o que en un caso peor, se elija un significado incorrecto.

2. Supóngase que el usuario tiene seleccionadas las siguientes palabras en su perfil:

- *Animals*
- *Sports*

Con la información de sus significados e hipónimos se arma el *Conjunto Perfil*:

{*acrodont, adult, and, anim, aquat, archeri, athlet, basebal, basketbal, be, beast, bipe, blood, box, brute, captiv, chordat, climb, clown, comedi, conceptu, contact, crawli, creatur, creepi, critter, cycl, darter, dead, domest, drolleri, ectotherm, egg, embryo, equit, exercis, fauna, femal, fertil, fiction, field, footbal, freak, fun, funni, game, giant, golf, gymanst, gymnast, herbivor, horseback, insectivor, invertebr, jocos, jocular, judo, larva, lusu, male, marin, mate, metazoan, migrat, molter, monster, monstros, moulter, mutant, mutat, natura, offspr, omnivor, outdoor, paronomasia, peeper, pet, plai, pleurodont, poikilotherm, predat, predatori, prei, profession, pun, pureblood, purebr, quarri, race, racer, rang, ride, rock, row, scaveng, sea, skate, ski, sled, spectat, sport, stayer, stunt, sumo, survivor, team, tenni, thoroughbr, track, variat, varment, varmint, vermin, waggeri, waggish, water, wordplai, work, wrestl, young, zooplankton*}

Se busca el término "*bat*", que tiene 5 sentidos como sustantivo:

- bat, chiropteran -- (nocturnal mouselike mammal with forelimbs modified to form membranous wings and anatomical adaptations for echolocation by which they navigate)
- bat, at-bat -- ((baseball) a turn batting; "he was at bat when it happened"; "he got 4 hits in 4 at-bats")
- squash racket, squash racquet, bat -- (a small racket with a long handle used for playing squash)
- cricket bat, bat -- (a bat used in playing cricket)
- bat -- (a club used for hitting a ball in various games)

Figura 4.3 Sentidos otorgados por WordNet al sustantivo "*bat*"

Como no se encuentran coincidencias entre el *Conjunto Perfil* y los *Conjuntos Definición*, se obtienen los hiperónimos para cada definición y se agregan a cada *Conjunto Definición*.

Los *Conjuntos Definición* quedan conformados de la siguiente manera:

- {*anim, bat, be, beast, brute, chiropteran, chordat, craniat, creatur, entiti, eutherian, fauna, live, mammal, object, organ, physic, placent, thing, vertebr*}
- {*act, action, activ, at, bat, human, plai, turn*}
- {*artefact, artifact, bat, entiti, implement, instrument, object, physic, racket, racquet, sport, squash, thing, unit, whole*}
- {*artefact, artifact, bat, cricket, entiti, equip, good, instrument, object, physic, sport, thing, unit, whole*}
- {*artefact, artifact, bat, club, entiti, implement, instrument, object, physic, stick, thing, unit, whole*}

Ahora aparece una coincidencia con la definición número 1, ya que el murciélago ("*chiropteran*") es un animal (aparece *anim*). También hay una coincidencia con las definiciones número 3 y 4, porque ambas

contienen la palabra *sport*. Aquí se llega a un estado en el que no es posible decidir cual sentido es conveniente para el usuario.

Una opción para resolver la ambigüedad es ofrecerle al usuario los sentidos con mayor *Valor de Concordancia* para que él decida.

Sin embargo, existe un problema más grave. Por más que el usuario determine un conjunto de intereses que realmente refleje sus deseos, puede suceder que en determinadas circunstancias, este quiera buscar información relacionadas con otras temáticas.

Volviendo al primer ejemplo, cuando este ingresa la consulta "*groups*", podría haber estado interesado en obtener información sobre diferentes grupos socio-culturales y no sobre grupos matemáticos.

Este enfoque resulta cómodo ya que son pocas las situaciones es que es necesaria la interacción, pero suele cometer muchos errores dejando descontento al usuario.

4.3.3 Expansión utilizando historial de consultas y perfil de usuario

Un intento de solucionar el cambio de intereses de los usuarios consiste en manejar las consultas realizadas previamente como información para obtener el sentido de los términos ingresados. Para esto se agregan las raíces de los términos de las últimas *N* consultas al *Conjunto Perfil* y se calculan los *Valores de Concordancia* de la misma forma que para el método anterior.

La metodología resulta efectiva cuando, por ejemplo, el usuario antes de buscar el término "*bat*" realizó una búsqueda relativa al deporte *baseball*, en ese caso, va a tener un mayor *Valor de Concordancia* la definición número 3, lo cual soluciona la ambigüedad.

Sin embargo, el método no funciona correctamente cuando el usuario realiza las primeras consultas sobre determinada temática. Tampoco funciona cuando se pasa rápidamente de un área de conocimiento a otra.

4.3.4 Expansión utilizando ancestros de ODP

Una opción no manejada hasta ahora por la gente que se dedica al área de *Recuperación de Información*, es la expansión de consultas utilizando términos relacionados extraídos del *Open Directory Project* (*ODP*)

El *ODP* es una ontología definida por humanos. Esta tiene algunas características interesantes:

- Intenta cubrir las diferentes áreas de información que se encuentran en la *web*.
- Es la más grande escrita por humanos.
- Es actualizada semanalmente con nuevas categorías.

La idea es encontrar apariciones de los términos que el usuario ingresó en las categorías de la ontología. A partir de esto, se muestra un árbol con las categorías encontradas, donde el usuario podrá elegir otras categorías que le parezcan relevantes respecto de la información que desea encontrar.

Una vez elegidas las categorías de interés. El sistema expande la consulta con ancestros de las categorías. En algunos casos, esto funciona correctamente, pero en otros resulta en una pérdida de precisión de la consulta realizada.

UN EJEMPLO

Buscando información sobre comida para perros, el usuario ingresa la palabra "*dog*", obteniendo los resultados que aparecen en la Figura 4.4.

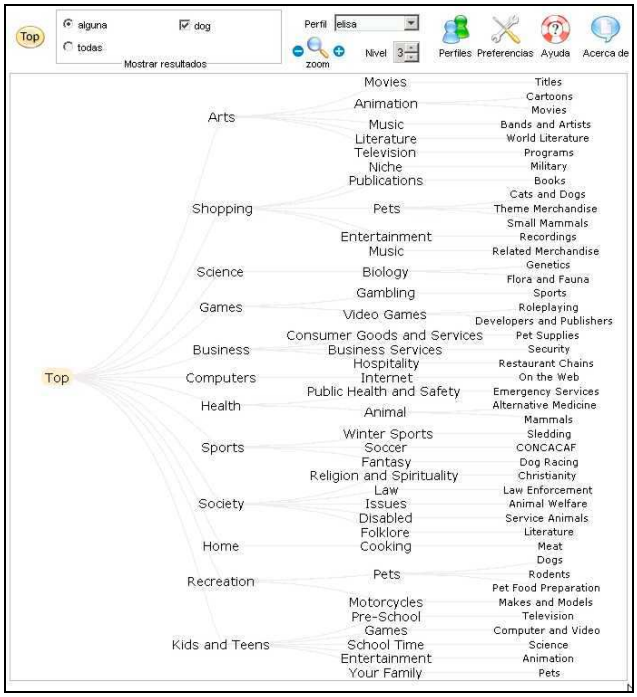

Figura 4.4 Categorías relacionadas con "dog"

Luego selecciona la categoría "*Meat*", como se puede ver en la Figura 4.5.

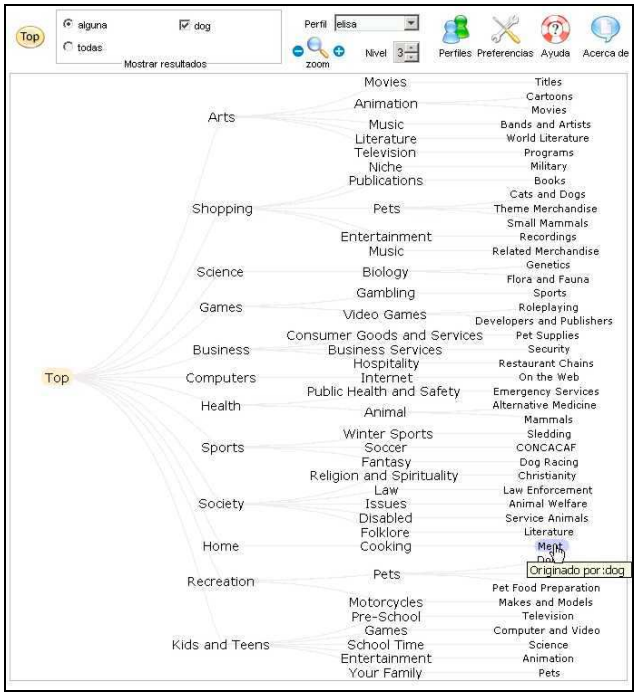

Figura 4.5 Selección de categorías relacionadas con "dog"

En caso de expandir la consulta con la palabra *cooking* se estaría perdiendo precisión, y más aún, la consulta se estaría alejando de los deseos del usuario.

Se podría utilizar otro ancestro en la jerarquía de categorías pero los resultados no parecen ser mejores.

4.3.5 Expansión utilizando definiciones de ODP

Existe un trabajo en el área que intenta detectar el significado de los términos que aparecen en la ontología *ODP*. Esta desambiguación se realiza utilizando *WordNet* y los ancestros de cada palabra del árbol. Con un método similar al descrito en el ítem 3.3.2, desambigua (cuando considera tener información para hacerlo) los sentidos de los términos del árbol [Sant03].

Esta metodología tiene la ventaja de que es posible realizar un preprocesamiento donde a cada categoría de *ODP* se le asigna el sentido elegido por el sistema.

Se podría utilizar la metodología agregando a la consulta, sinónimos de los sentidos de las categorías elegidas por el usuario. Sin embargo, esto tiene la desventaja de no poder utilizar las demás categorías que eligió ni los otros términos que ingresó para realizar la desambiguación.

4.3.6 Utilización de stemmers y lematizadores

Un *stemmer* es un sistema que obtiene la forma de una palabra luego de que todos sus afijos (prefijos y sufijos) fueron removidos (por una explicación más específica dirigirse al Capítulo 2: Definiciones Básicas).

Un lematizador es un sistema que obtiene una palabra en su forma más simple. Esta es la que aparece como entrada en diccionarios y tesauros (ver Capítulo 2).

El objetivo de utilizar estos sistemas es: obtener la raíz de una palabra y no quedarse con morfemas flexivos o derivativos que no se utilizan en las ontologías manejadas.

Se estudió la utilización del algoritmo de *Porter* [Port80] para este problema. Sin embargo, se encontraron algunas fallas que descartaron su ingreso al sistema para la tarea de obtener la raíz de un término y luego buscar los resultados relacionados en la ontología *ODP*. Por tanto, se decidió utilizar el lematizador que trae *WordNet* para realizar esta tarea. Al tener pensando utilizar la herramienta, no resultaba contraproducente tomar esta decisión.

4.4 Descripción

Como ya se comentó anteriormente, el sistema se basa en buscar términos relacionados con una consulta inicial, para realizar una expansión con estos conceptos.

El usuario ingresa una consulta (secuencia de palabras) en el sistema. Luego, el programa busca apariciones de los términos que el usuario ingresó en las categorías de la ontología.

Por ejemplo:

Se ingresa: "*green apples*" y el sistema encuentra todos los caminos de la jerarquía (ontología) que contiene el lema de alguna de las dos palabras.

Entre ellos:

Top/Arts/Movies/Genres/Romance/Book_Adaptations/Carmen_Green
Top/Arts/Movies/Titles/A/Apple,_The_-_1980
Top/Arts/Movies/Titles/B/Big_Green,_The
Top/Arts/Movies/Titles/F/Fried_Green_Tomatoes
Top/Arts/Movies/Titles/G/Green_Dragon,_The
Top/Arts/Movies/Titles/G/Green_for_Danger
Top/Arts/Movies/Titles/G/Green_Hornet,_The
Top/Arts/Movies/Titles/G/Green_Mile,_The
Top/Arts/Movies/Titles/H/How_Green_Was_My_Valley
Top/Arts/Movies/Titles/S/Soylent_Green
Top/Computers/Companies/Product_Support/Apple
Top/Computers/Graphics/Clip_Art/Apple_Logos
Top/Computers/Hardware/Peripherals/Audio/Portable_Players/Apple_iPod
Top/Computers/Systems/Apple
Top/Computers/Systems/Apple/Apple_II
Top/Home/Cooking/Baking_and_Confections/Pies_and_Pastry/Apple_Pies
Top/Home/Cooking/Beverages/Apple_Juice_and_Cider
Top/Home/Cooking/Fruits_and_Vegetables/Apples
Top/Home/Cooking/Fruits_and_Vegetables/Apples/Apple_Dumplings
Top/Home/Cooking/Fruits_and_Vegetables/Greens
Top/Home/Cooking/Soups_and_Stews/Fruit_and_Vegetable/Greens
Top/Shopping/Food/Confectionery/Candy_Apples
Top/Shopping/Food/Produce/Fruit/Apples
Top/Shopping/Health/Nutrition/Green_Foods
Top/Shopping/Niche/Green_Living
Top/Shopping/Sports/Football/American/NFL/Green_Bay_Packers

Figura 4.6 Algunos resultados de *ODP* para la consulta "*green apples*"

A partir de esto, se genera un árbol con las categorías encontradas, donde el usuario podrá elegir categorías que le parezcan relevantes con respecto a la información que desea encontrar. Además puede buscar categorías más específicas que surjan a partir de una elegida.

Siguiendo con el ejemplo, se pueden buscar las categorías debajo de:

Top/Computers/Systems/Apple

Figura 4.7 Categoría elegida para buscar sus descendientes

Obteniendo:

Top/Computers/Systems/Apple/Apple_II
Top/Computers/Systems/Apple/Macintosh
Top/Computers/Systems/Apple/Lisa
Top/Computers/Systems/Apple/History
Top/Computers/Systems/Apple/Personal_Pages

Figura 4.8 Descendientes obtenidos

En algunos casos, por ejemplo cuando se ingresan demasiadas palabras al sistema, o cuando la cantidad de categorías relacionadas con un término, es demasiado grande, sucede que es difícil visualizar correctamente la información. Para estas situaciones, el sistema ofrece la posibilidad de mostrar las categorías relacionadas con cualquier subconjunto de las palabras ingresadas. Incluso puede mostrar únicamente aquellas que aparecen como relacionadas para todas las palabras de un subconjunto dado.

Por ejemplo:

Se desea observar solo las relacionadas con "*green*", y aparecen:

Top/Arts/Movies/Genres/Romance/Book_Adaptations/Carmen_Green
Top/Arts/Movies/Titles/B/Big_Green,_The
Top/Arts/Movies/Titles/F/Fried_Green_Tomatoes
Top/Arts/Movies/Titles/G/Green_Dragon,_The
Top/Arts/Movies/Titles/G/Green_for_Danger
Top/Arts/Movies/Titles/G/Green_Hornet,_The
Top/Arts/Movies/Titles/G/Green_Mile,_The
Top/Arts/Movies/Titles/H/How_Green_Was_My_Valley
Top/Arts/Movies/Titles/S/Soylent_Green
Top/Home/Cooking/Fruits_and_Vegetables/Greens
Top/Home/Cooking/Soups_and_Stews/Fruit_and_Vegetable/Greens
Top/Shopping/Health/Nutrition/Green_Foods
Top/Shopping/Niche/Green_Living
Top/Shopping/Sports/Football/American/NFL/Green_Bay_Packers

Figura 4.9 Algunos resultados de *ODP* relacionados con "*apples*"

O se quiere ver las que están relacionadas con "*apple*" y con "*green*" a la vez, teniendo como resultado la visualización de:

Top/Arts/Movies/Titles
Top/Home/Cooking/Fruits_and_Vegetables

Figura 4.10 Algunos resultados de *ODP* relacionados con "*apples*" y con "*green*"

También se puede acercar y alejar la representación del árbol de categorías para cuando se quiere trabajar con todos los datos.

Una vez seleccionadas las categorías deseadas, se utiliza un diccionario (en este caso *WordNet*) para intentar descifrar el verdadero sentido de los términos de la consulta.

Para esto, se utiliza la siguiente información:

- Consulta ingresada.
  - Sentido de cada palabra.
  - Sinónimos.
  - Hiperónimos (términos más generales).
  - Restantes palabras de la consulta.
- Datos de la jerarquía.
  - Categorías seleccionadas.
  - Ancestros de las categorías seleccionadas.

Con esa información se busca el sentido que más se ajuste a cada término.

En caso de obtener un sentido, se muestran sinónimos e hiperónimos relativos a cada palabra desambiguada.

A continuación, se puede especificar aún más la consulta eligiendo términos relacionados con las palabras iniciales para los sentidos obtenidos.

En caso de elegir sinónimos de los términos el sistema expandirá la consulta, es decir, la generará de forma tal que sea posible encontrar documentos que tengan uno de los sinónimos sin necesidad de que aparezcan los demás.

Si se eligen otros términos relacionados, el sistema precisará la consulta, generándola de manera que sean más relevantes aquellos documentos que tengan todos los términos elegidos.

Al final, el programa creará la consulta para el buscador elegido (Por ejemplo: *Google*, *Yahoo* o *Altavista*) que contiene todos los términos seleccionados (dispuestos de la manera comentada) y algunos agregados cuando se lo considera conveniente.

Resulta favorable agregar términos automáticamente a la consulta cuando se considera que se puede desambiguar (con alta probabilidad de acierto) un término elegido del árbol de categorías. En este caso, se añade un sinónimo para expandir la consulta.

Este enfoque no ha sido aplicado anteriormente, por lo que resulta interesante como nueva dirección a escoger.

4.5 Ejemplos

A continuación se presenta una serie de ejemplos para intentar aclarar el funcionamiento del programa y mostrar su desempeño en diferentes situaciones.

4.5.1 Visualización

Se pasa al sistema la consulta "*art body work*".

Como se puede observar en la Figura 4.11, la cantidad de resultados retornados es abrumadora.

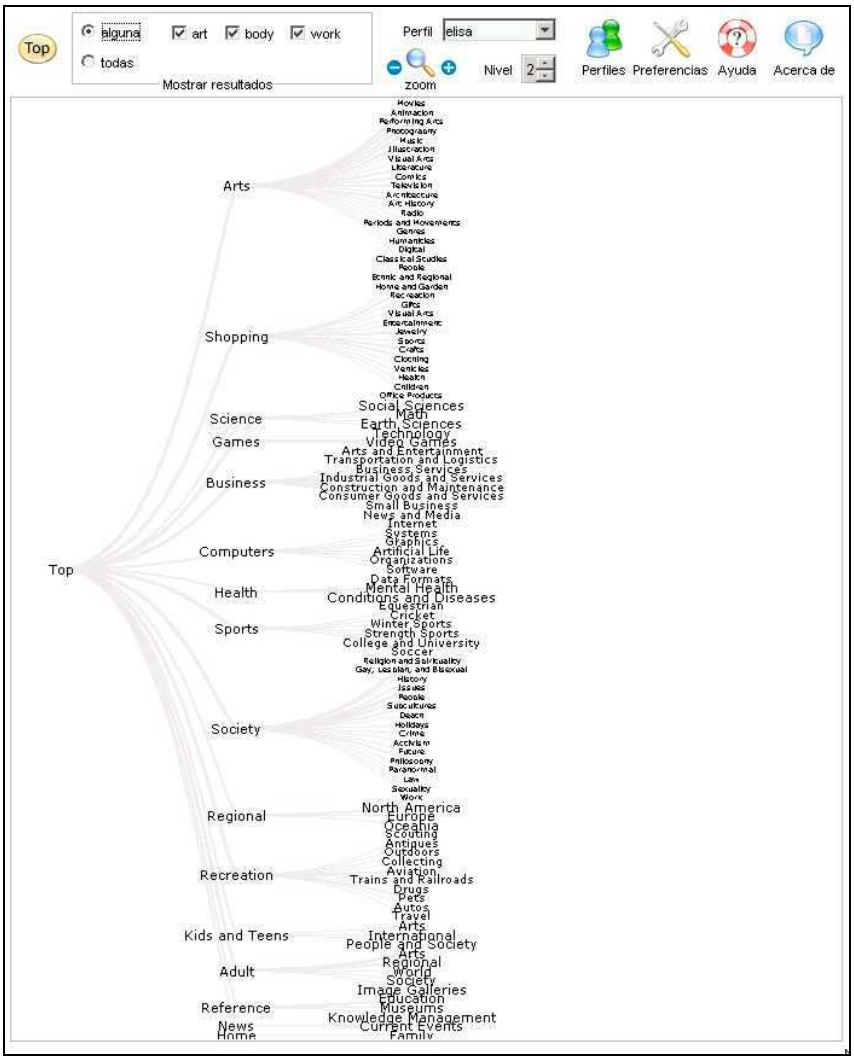

Figura 4.11 Resultados en exceso

Para esto, el sistema permite ampliar y disminuir la superficie de visualización (Figura 4.12).

Figura 4.12 Ampliación de Resultados

O mostrar menos niveles de la jerarquía seleccionando un valor menor en el campo "*Level*" (Figura 4.13).

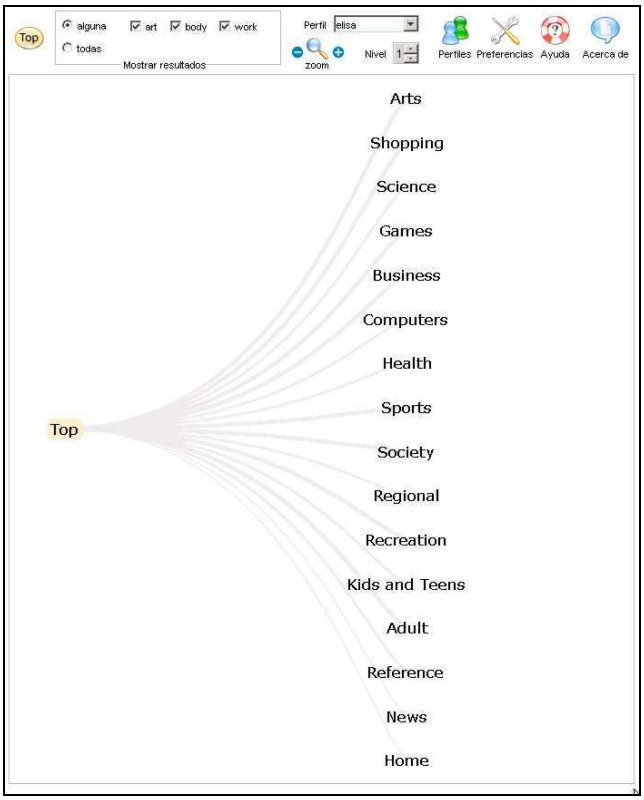

Figura 4.13 Niveles de visualización

También ofrece la posibilidad de exponer únicamente las categorías del árbol que están relacionadas con todas las palabras ingresadas (Figura 4.14).

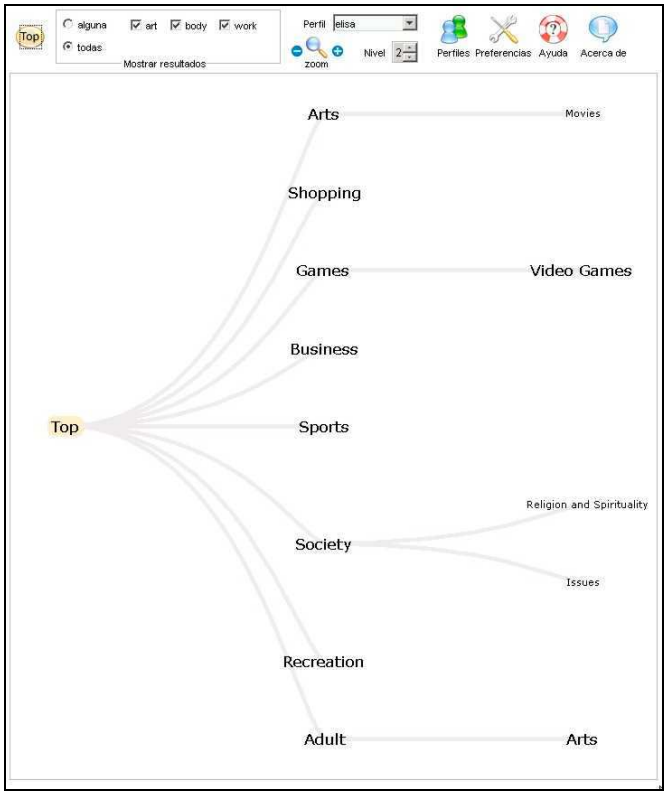

Figura 4.14 Filtros de categorías

O las que aparecen relacionadas con algunas palabras dentro de un subconjunto del conjunto inicial (Figura 4.15).

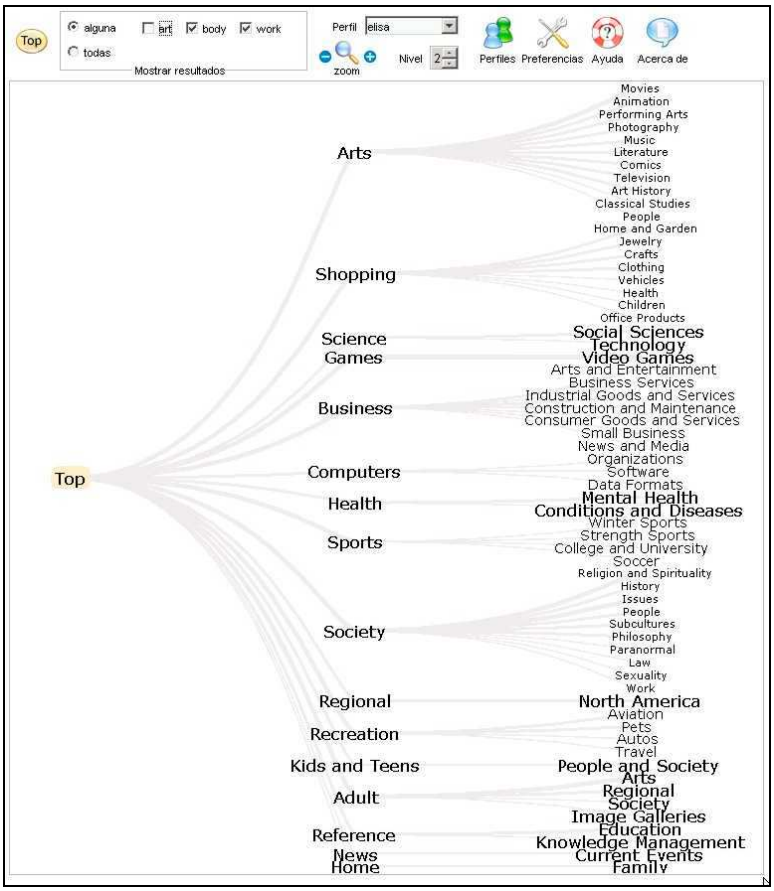

Figura 4.15 Filtros de categorías

Esta desarrollada también la funcionalidad de ofrecer una lista de las categorías "hojas" del árbol relacionadas con los resultados. Entonces, si se quiere obtener todas las categorías relacionadas con las palabras ingresadas a través de la categoría "Shopping" se debe presionar el botón derecho sobre esta, obteniendo los resultados que aparecen en la Figura 4.16.

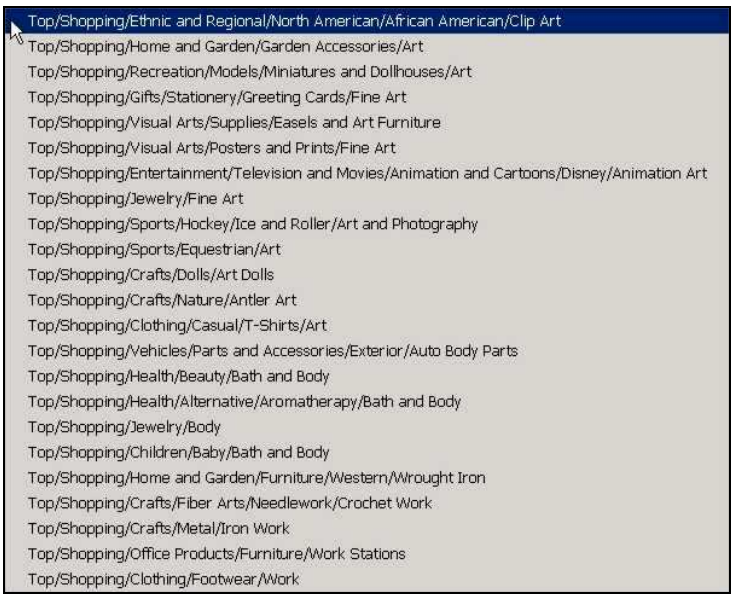

Figura 4.16 Categorías relacionadas con "art body work" a partir de la rama derivada de "Shopping"

En caso de estar filtrando los resultados para algunas de las palabras ingresadas. La lista de categorías relacionadas se va a modificar. Por ejemplo: eligiendo únicamente los resultados relacionados con el término "work" se obtienen las siguientes categorías relacionadas (Figura 4.17):

Figura 4.17 Categorías relacionadas con "work" a partir de la rama derivada de "Shopping"

Además se permite mostrar categorías más específicas para una categoría elegida. Como se puede observar en la Figura 4.18, la categoría "*Social Work*" no tiene predecesores, sin embargo presionando sobre ella se obtienen las categorías que se pueden distinguir en la Figura 4.19.

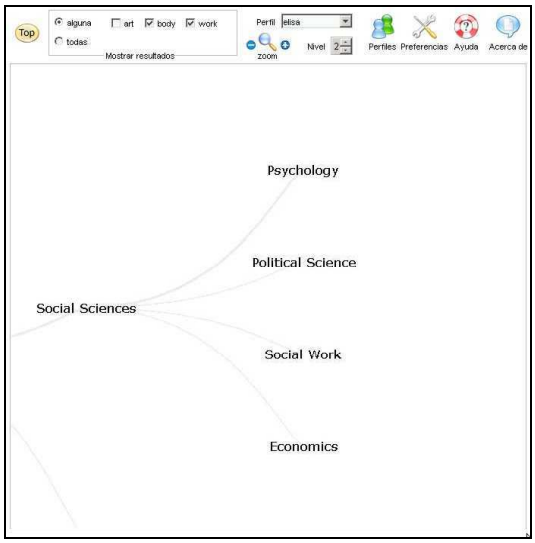

Figura 4.18 Categorías iniciales

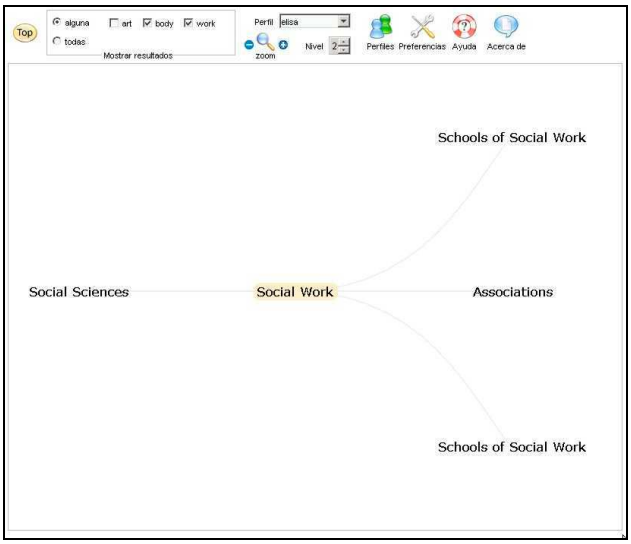

Figura 4.19 Obtención de más categorías

Todas estas opciones ayudan a visualizar la información y poder elegir las categorías que se relacionen más estrechamente con los conceptos buscados.

A continuación se marcan las categorías identificadas como relevantes (Figura 4.20), en este caso únicamente:

- Art History

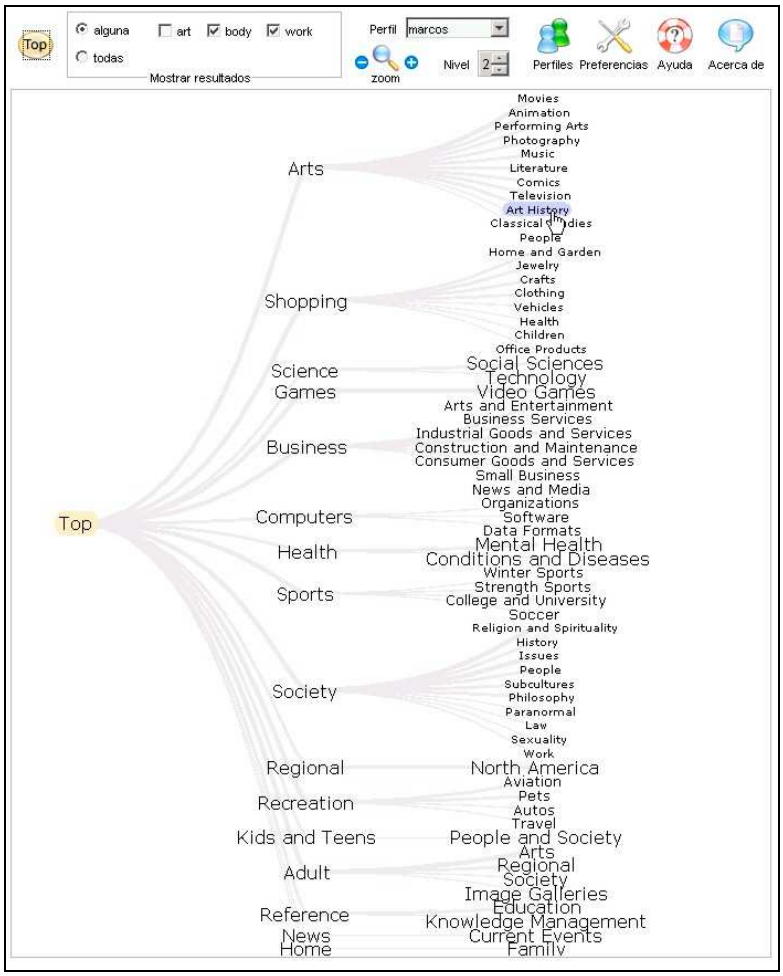

Figura 4.20 Selección de categorías

Y se solicita al sistema que encuentre la definición para las palabras ingresadas.

Aquí se obtiene la definición encontrada para cada término (Figura 4.21, Figura 4.22 y Figura 4.23):

- *art*: fine art -- (the products of human creativity; works of art collectively; "an art exhibition"; "a fine collection of art").

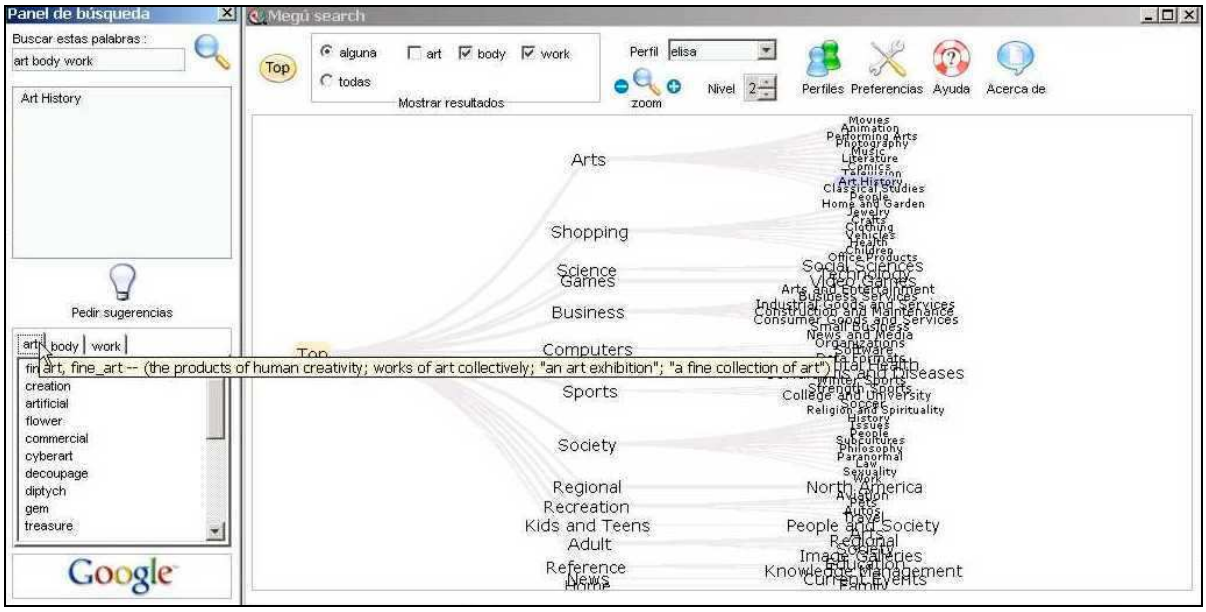

Figura 4.21 Desambiguación de "art"

Se puede observar además la lista de palabras relacionadas ofrecidas para la selección.

- *body*: (a group of persons associated by some common tie or occupation and regarded as an entity; "the whole body filed out of the auditorium").

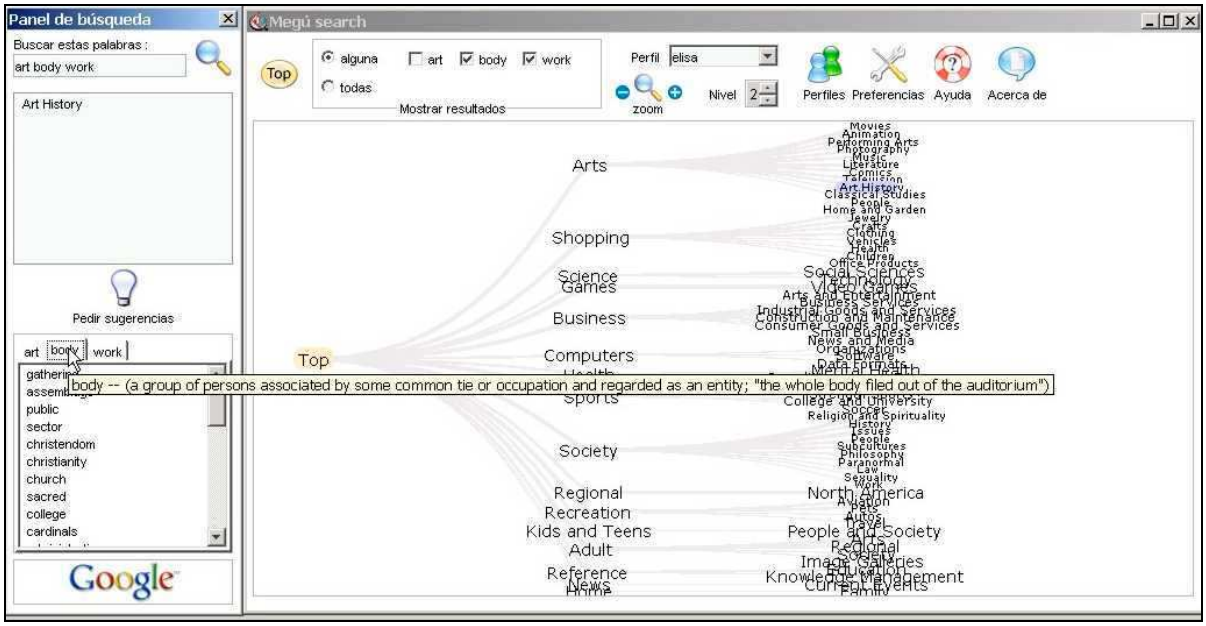

Figura 4.22 Desambiguación de "body"

- *work*: oeuvre, body of work -- (the total output of a writer or artist (or a substantial part of it); "he studied the entire Wagnerian oeuvre"; "Picasso's work can be divided into periods").

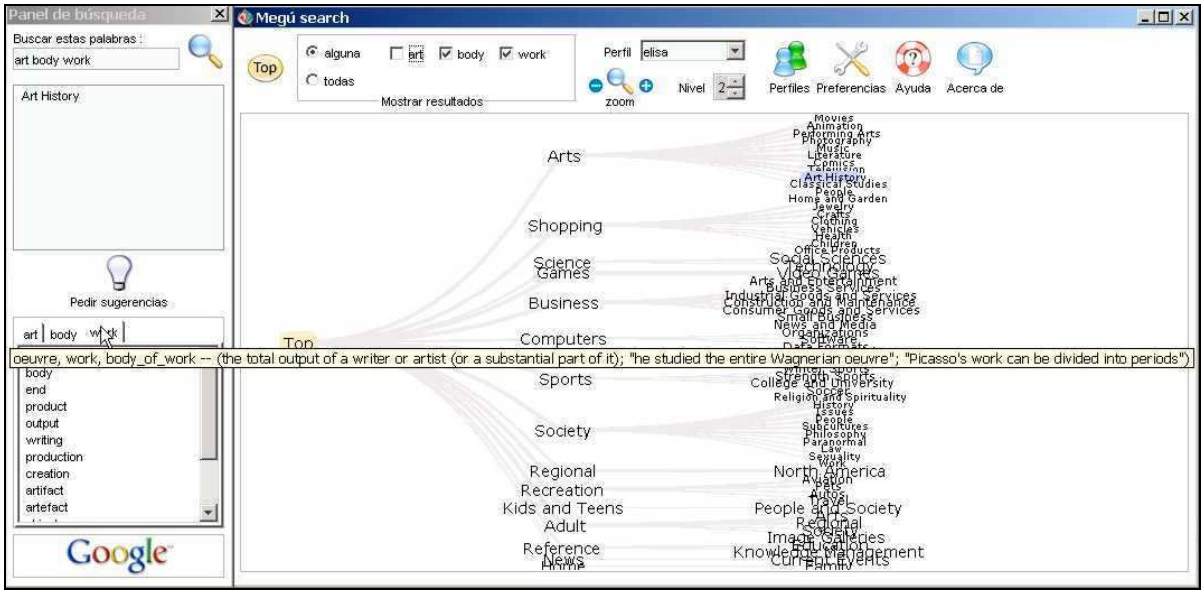

Figura 4.23 Desambiguación de "work"

Una vez elegidos los términos deseados se pide la generación de la consulta apretando en el botón con la etiqueta "*Google*".

Este cual solicita una instancia del explorador *web* que tenga al sistema, y llama al buscador elegido con la consulta generada. Para modificar el navegador se debe presionar el botón derecho del ratón sobre el botón con la etiqueta "*Google*". En el cliente desarrollado se ofrecen los buscadores: *Google*, *Yahoo*; *Altavista.*

4.5.2 Consulta: "Apple"

Si se ingresa la consulta "*apple*" se obtiene un árbol con los resultados mostrados en la Figura 4.24.

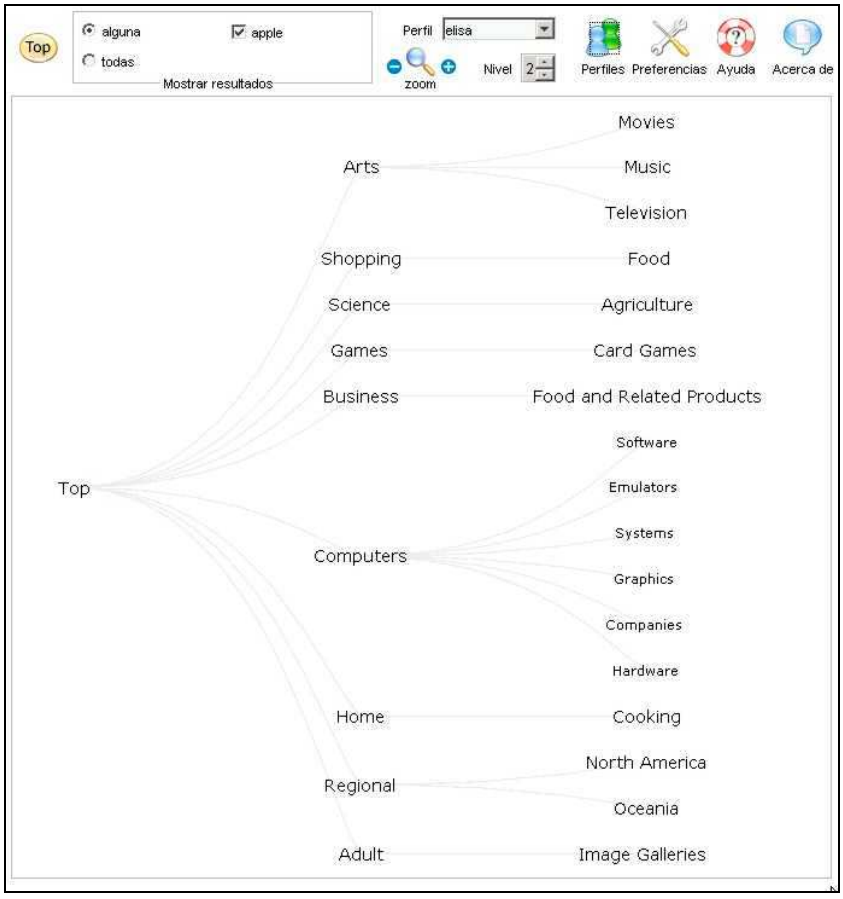

Figura 4.24 Categorías relacionadas con "apple"

Aquí se puede elegir cualquier categoría (nodo del árbol). Estas serán utilizadas para la expansión. Además se usarán para intentar descifrar el sentido de las palabras de la consulta inicial a realizarse de la siguiente manera:

- Se busca el sentido en el diccionario (en este caso *WordNet*).
- Si existe más de una definición para la palabra, se intenta descifrar el requerido a partir de las palabras elegidas en la ontología.
- Si se encuentra un significado apropiado, se ofrecen términos relacionados para continuar la expansión de la consulta.

El usuario puede entonces, elegir más términos para expandir la consulta.

Se supone que el usuario elige el siguiente término en el árbol:

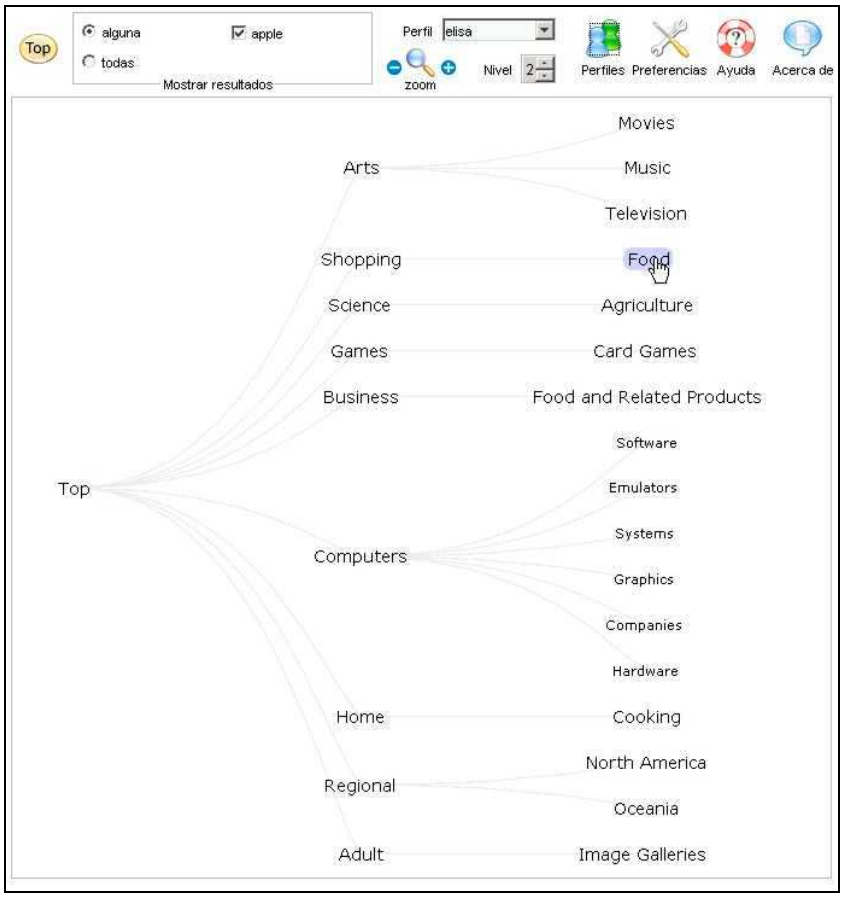

Figura 4.25 Selección de categorías relacionadas con "apple"

El sistema busca la palabra "*apple*" en el diccionario, obteniendo 2 resultados:

- apple -- (fruit with red or yellow or green skin and sweet to tart crisp whitish flesh)
- apple, orchard apple tree, Malus pumila -- (native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits)

Entonces el sistema intenta desambiguar la palabra a partir de las selecciones en el árbol y obtiene como resultado el sentido número 1. A continuación busca el conjunto de sinónimos e hiperónimos para ofrecer al usuario. Obteniendo:

{*edible, fruit, pome, false, orchard, tree, malus, pumila, crab, crabapple, eating, dessert, cooking, produce, green, goods, groceries, garden, truck, food, solid, substance, matter, reproductive, structure, plant, organ, part*}

Si se elige el término *fruit*, y se manda a buscar, se obtiene la siguiente consulta: "*apple food fruit*".

Como se puede observar, el orden en que los términos son agregados a la consulta esta dado por:

- Palabras digitadas por el usuario.
- Palabras seleccionadas en el árbol.
- Palabras seleccionadas en el conjunto de sugerencias.

4.5.3 Consulta: "Bank"

Si se ingresa la consulta "*bank*" se obtiene un árbol con los resultados de la Figura 4.26.

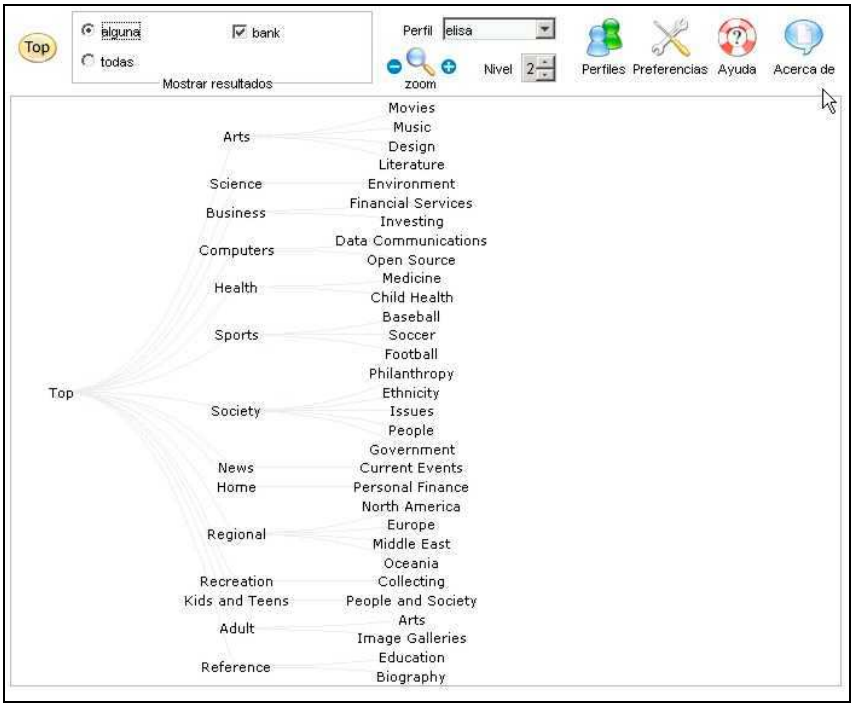

Figura 4.26 Categorías relacionadas con "bank"

Se supone que el usuario elige el siguiente término del árbol:

- *Investing*

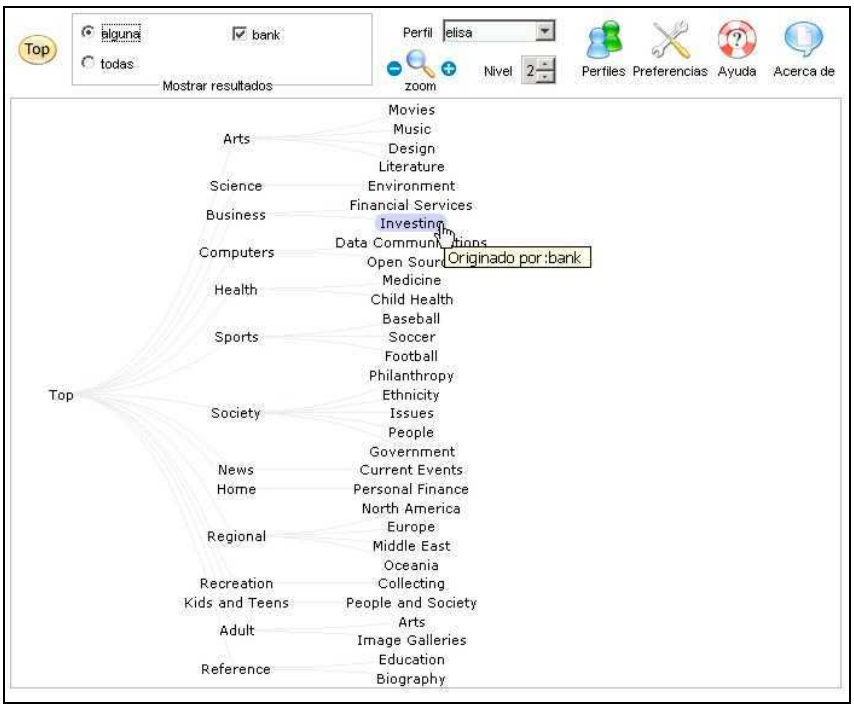

Figura 4.27 Selección de categorías relacionadas con "bank"

A continuación se busca la palabra "*bank*" en el diccionario, y se obtienen resultados en dos categorías gramaticales diferentes: como verbo y como sustantivo. Para la categoría verbal hay 8 sentidos:

- bank -- (tip laterally; "the pilot had to bank the aircraft")
- bank -- (enclose with a bank; "bank roads")
- bank -- (do business with a bank or keep an account at a bank; "Where do you bank in this town?")
- bank -- (act as the banker in a game or in gambling)
- bank -- (be in the banking business)
- deposit, bank -- (put into a bank account; "She deposites her paycheck every month")
- bank -- (cover with ashes so to control the rate of burning; "bank a fire")
- trust, swear, rely, bank -- (have confidence or faith in; "We can trust in God"; "Rely on your friends"; "bank on your good education"; "I swear by my grandmother's recipes")

Y como sustantivo se encontraron 10 definiciones:

- depository financial institution, bank, banking concern, banking company -- (a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home")
- bank -- (sloping land (especially the slope beside a body of water); "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents")
- bank -- (a supply or stock held in reserve for future use (especially in emergencies))
- bank, bank building -- (a building in which commercial banking is transacted; "the bank is on the corner of Nassau and Witherspoon")
- bank -- (an arrangement of similar objects in a row or in tiers; "he operated a bank of switches")
- savings bank, coin bank, money box, bank -- (a container (usually with a slot in the top) for keeping money at home; "the coin bank was empty")
- bank -- (a long ridge or pile; "a huge bank of earth")
- bank -- (the funds held by a gambling house or the dealer in some gambling games; "he tried to break the bank at Monte Carlo")
- bank, cant, camber -- (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- bank -- (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning); "the plane went into a steep bank")

Entonces el sistema intenta desambiguar la palabra a partir de las selecciones en el árbol y obtiene como resultado el sentido número 3.

- bank -- (do business with a bank or keep an account at a bank; "Where do you bank in this town?")

Esto resulta satisfactorio porque se encontró una definición que se ajusta a los deseos del usuario. Entonces se busca el conjunto de sinónimos e hiperónimos para ofrecer al usuario. Obteniendo los resultados de la Figura 4.28.

Figura 4.28 Sugerencias encontradas para "bank"

Si se elige el término: *transact*, y se manda a buscar, se obtiene la siguiente consulta: "*bank (investing OR investment) transact*"

Como se puede observar, apareció el término *investment*, que fue agregado automáticamente por el sistema, ya que consideró que el significado de la palabra *investing*, obtenida del árbol es:

- investing, investment -- (the act of investing; laying out money or capital in an enterprise with the expectation of profit)

Donde los significados posibles eran:

Como verbo:

- invest, put, commit, place -- (make an investment; "Put money into bonds")
- endow, indue, gift, empower, invest, endue -- (give qualities or abilities to)
- invest, clothe, adorn -- (furnish with power or authority; of kings or emperors)
- invest, vest, enthrone -- (provide with power and authority; "They vested the council with special rights")
- induct, invest, seat -- (place ceremoniously or formally in an office or position; "there was a ceremony to induct the president of the Academy")

Y como sustantivo:

- investing, investment -- (the act of investing; laying out money or capital in an enterprise with the expectation of profit)

La selección del significado está dada por la concordancia entre los hiperónimos de ese sentido y las palabras ingresadas por el usuario (en este caso "*bank*"). Al elegir un significado pasa entonces a elegir un sinónimo de la palabra para ese sentido, expandiendo automáticamente la consulta.

Supóngase ahora que se eligieron estas categorías del árbol:

- *Financial Services*
- *Personal Finance*

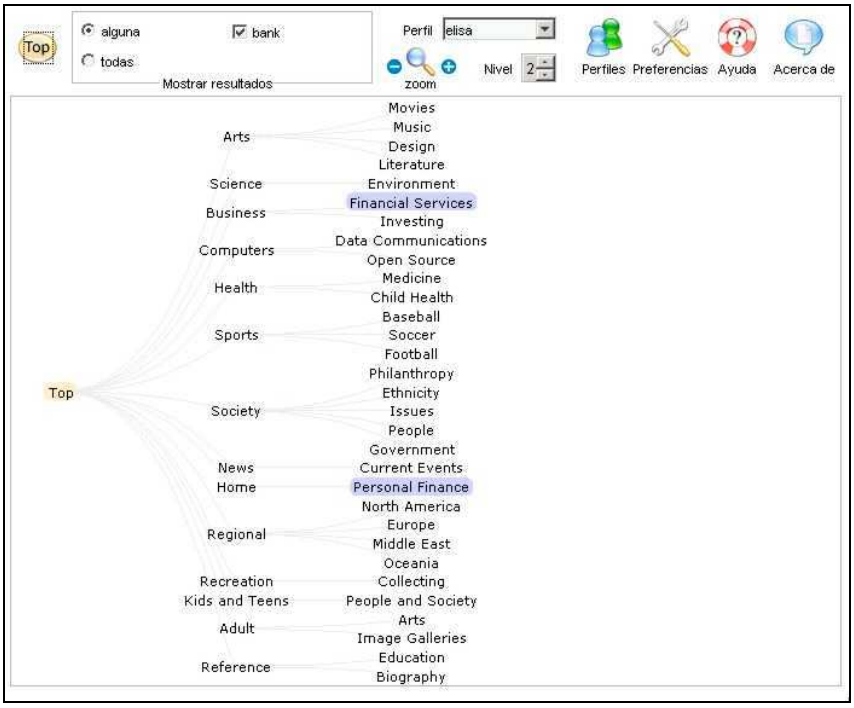

Figura 4.29 Nueva selección de categorías relacionadas con "bank"

Ahora el sentido elegido por el sistema es el siguiente:

- depository financial institution, bank, banking concern, banking company -- (a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home")

Obteniendo estos términos para la expansión:

{*depository, financial, institution, banking, concern, company, organization, organisation, industry, system, credit, union, federal, reserve, agent, commercial, full, service, state, lead, member, merchant, acquirer, thrift, home, loan, establishment, social, group, grouping*}

4.5.4 Consulta: "Jaguar"

Si se ingresa la consulta "*jaguar*" se obtiene un árbol con los siguientes resultados:

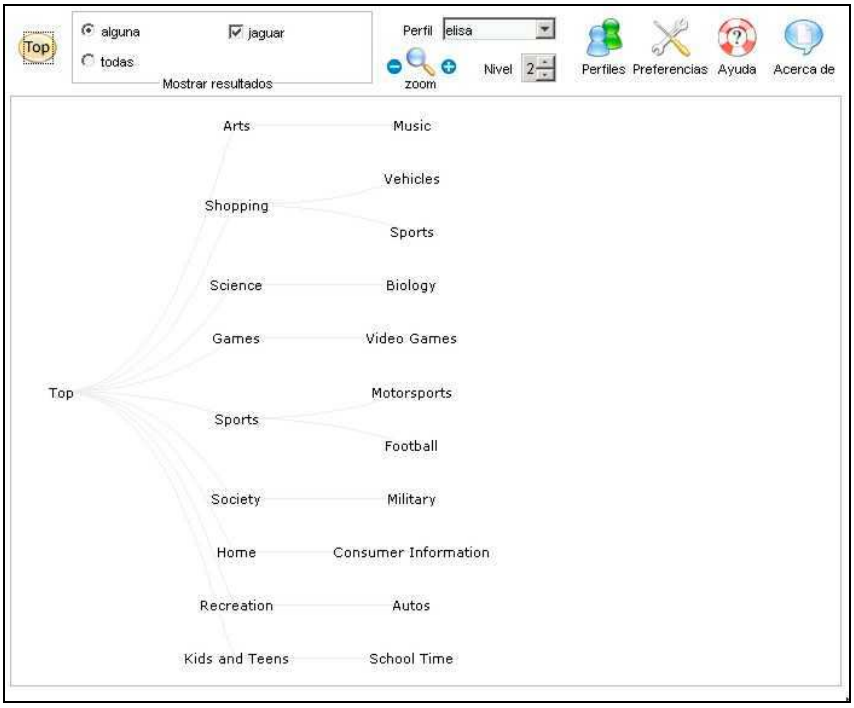

Figura 4.30 Categorías relacionadas con "jaguar"

Se supone que el usuario elige el siguiente término en el árbol:

- *Biology*

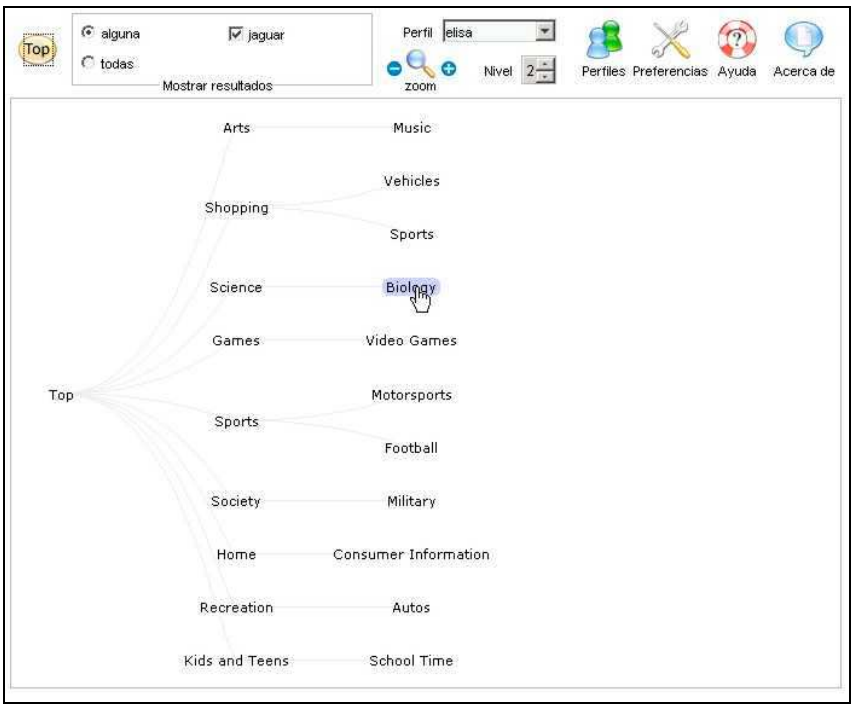

Figura 4.31 Selección de categorías relacionadas con "jaguar"

El sistema busca la palabra "*jaguar*" en el diccionario, y se obtiene el siguiente resultado:

- jaguar, panther, Panthera onca, Felis onca -- (a large spotted feline of tropical America similar to the leopard; in some classifications considered a member of the genus Felis)

Entonces el sistema utiliza ese sentido y busca el conjunto de sinónimos e hiperónimos para ofrecer al usuario:

{*panther, panthera, onca, felis, big, cat, genus, feline, felid, carnivore, placental, mammal, eutherian*}

Figura 4.32 Sugerencias encontradas para "jaguar"

Si se elige el término: *onca*, y se manda a buscar, se obtiene la siguiente consulta: "*(jaguar OR onca) (biology OR biological)*".

Como se puede observar, aparece el término: *biological*, agregado automáticamente por el sistema, ya que consideró que el significado de la palabra *biology* es:

- biology, biological science -- (the science that studies living organisms)

Este fue elegido entre:

- biology, biological science -- (the science that studies living organisms)
- biology -- (characteristic life processes and phenomena of living organisms; "the biology of viruses")
- biota, biology -- (all the plant and animal life of a particular region)

Esto se debe a la aparición del término *science* en la rama del árbol elegida por el usuario. Entonces, se agrega el término *biological* a la consulta.

Además, como el término elegido *onca* aparece en la propia definición de *jaguar*, esta es agregada con el operador *or* a la consulta porque se puede entender como sinónimo. De esta manera se expande el conjunto de documentos sobre el cual se busca, porque no solo aparecerán aquellos que tienen el término *jaguar* sino también aquellos que contengan *onca*.

4.6 Tecnologías utilizadas

Dadas los objetivos tan exigentes planteados, resulta conveniente y hasta casi necesario utilizar componentes y tecnologías preexistentes, ya que la complejidad del sistema hace imposible realizar una implementación desde cero.

Un componente de software es una pieza de código escrita anteriormente que define interfases y puede ser invocada para proveer una funcionalidad que el software encapsula.

Las tecnologías y componentes utilizadas se eligieron respetando ciertos puntos importantes para que la aplicación no pierda utilidad ni se reduzca su posibilidad de distribución. Estos ítems son los siguientes:

- Utilizar únicamente piezas de uso libre (gratis) y de código abierto (*OpenSource*).
- Evitar la necesidad de instalar aplicaciones específicas en la máquina que va a contenar al cliente de la aplicación.

Cumpliendo con estas condiciones se considera que la aplicación sigue siendo útil.

4.6.1 Tecnologías y componentes utilizados en el servidor:

A continuación se detallan las tecnologías y componentes que fueron utilizadas para la implementación del servidor de la aplicación.

WordNet:

Este es un sistema de referencia léxico *online*. Las formas de las palabras en *WordNet* están representadas en su ortografía usual. Los significados están representados en conjuntos de sinónimos (*Synsets*) que son listas de palabras intercambiables en algún contexto.

Se reconoce dos tipos de relaciones: léxicas y semánticas. Las relaciones léxicas se dan entre formas de palabras. Las relaciones semánticas se dan entre significados.

La existencia de este recurso lingüístico le ha abierto camino a muchos trabajos dentro de diferentes áreas. Por ejemplo, en Procesamiento de Lenguaje Natural (*Natural Language Processing – NLP*) se suele utilizar para desambiguar el sentido de una palabra. En general, en *Recuperación de Información* se utiliza para sugerir o agregar términos a una consulta.

En el trabajo presentado, *WordNet* se utiliza para:

- Encontrar el significado de los términos ingresados.
- Encontrar hiperónimos de los distintos sentidos de los términos ingresados para realizar la desambiguación de términos.
- Sugerir palabras relacionadas con cada término ingresado para expandir la consulta.

A consideración de Voorhees [Voor98], la utilización de recursos lingüísticos como *WordNet* para la expansión de consultas resultan efectivas si la cantidad de conceptos de esta es muy pequeña. En cambio, si la consulta tiene más conceptos, la utilidad desmejora hasta el punto de ser inútil este recurso.

En cambio Mandala [Mand98] sugiere que las expansiones de consultas utilizando recursos como *WordNet* puede llevar a una mejoría en la cantidad de documentos recuperados pero disminuye la precisión. Esto se debe a la gran cantidad de nombres propios que no maneja *WordNet*. Por citar un ejemplo *Apple* (la compañía *Apple Macintosh*) no se encuentra en el diccionario.

Las técnicas para expandir consultas son variadas, sin embargo la utilización de *WordNet* ha dado muy buenos resultados en el trabajo realizado ya que resultan muy útiles la gran cantidad de definiciones que contiene y las relaciones jerárquicas definidas entre los términos.

El sitio de *WordNet* es http://www.cogsci.princeton.edu/~wn/

JWNL:

Es una interfaz para la programación de aplicaciones (*Application Program Interface – API*) para *Java* generada para el procesamiento de diccionarios relacionales como *WordNet*. También provee de funcionalidades más allá del acceso a los datos, como por ejemplo descubrimiento de relaciones y procesamiento morfológico.

A partir de esta interfaz se pueden obtener todas las definiciones encontradas en *WordNet* para un término dado, así como sinónimos, hiperónimos, hipónimos, etc.

El sitio de desarrollo de *JWNL* es http://www.sourceforge.net/projects/jwordnet

MySQL:

Es el manejador de bases de datos (*Database Management System – DBMS*) gratis y de código abierto (*OpenSource*) para *SQL* más popular del mundo.

Este tiene ciertas características que lo hacen útil para el trabajo requerido. Consta de una confiable y extensible infraestructura para el desarrollo de aplicaciones de variado porte. Además, se han desarrollado interfases con prácticamente todos los lenguajes de programación actualmente utilizados lo que hace que los datos que sean guardados en esta clase de manejadores puedan ser accedidos desde casi cualquier aplicación desarrollada. Fue desarrollada junto con el manejador una extensa documentación, de fácil comprensión para quien ha trabajado con manejadores de bases de datos anteriormente.

El sitio de *MySQL* es http://www.mysql.com/

MySQL Connector/*Java*:

Es una implementación de la interfase para programación de aplicaciones (*API*) *JDBC* 3.0 de *Sun* para el manejador de bases de datos relacionales *MySQL*. Esta *API* permite establecer una conexión con el manejador relacional y provee de funciones para emitir las consultas necesarias en la base de datos generada para la aplicación.

El sitio de *MySQL Connector* es http://www.mysql.com/

ODP data dump:

Es el volcado de datos del conjunto de categorías del *Open Directory Project*. Esta es una ontología que intenta separar la información que se puede obtener en *Internet* en un conjunto de categorías.

Esta ontología está especificada en diferentes formatos, entre ellos: *XML* y texto plano.

Se eligió el pasaje de la información desde un archivo de texto plano hacia la base de datos por la simplicidad que este pasaje ofrecía. El formato del archivo ofrecido por el sitio es un listado de categorías jerárquicamente dispuestas y separadas por el carácter de fin de línea.

Un archivo de ejemplo puede ser el siguiente:

```
Top
Top/Arts
Top/Arts/Movies
Top/Arts/Movies/Hard_To_Kill
Top/Arts/Movies/Die_Hard
```

```
Top/Arts/Movies/Hard_Days
Top/Arts/Paintings
Top/Business
Top/Busines/Investing
```

Figura 4.33 EJemplo de formato del archivo de la ontología

Este archivo es ingresado a la base de datos a través de la aplicación para importar una ontología. El funcionamiento de esta aplicación está explicado en la Sección 4.10.1.

El sitio oficial es: http://www.dmoz.org/

Web Services:

Son aplicaciones para un entorno *web* que utilizan estándares basados en *XML* y protocolos de transporte para intercambiar datos con diferentes aplicaciones cliente. Estos generalmente se ajustan al protocolo estipulado por *SOAP* (explicado más adelante) para realizar el intercambio de información.

Java ofrece este tipo de servicios y se puede ver su especificación en:
http://developers.sun.com/techtopics/webservices/index.html

Web Services Description Language (WSDL):

Este es un formato *XML* para describir servicios *web* como un conjunto de funciones o funcionalidades que provee un servidor determinado. Las operaciones y mensajes son descritas de manera abstracta, y luego instanciadas en un protocolo de red y un formato de mensajes en concreto.

WSDL es extensible a la especificación del formato de mensajes y los protocolos de red a utilizar. Es posible utilizarlo con el protocolo *SOAP* explicado a continuación.

El sitio oficial es: http://www.w3.org/TR/wsdl

SOAP:

Es un protocolo simple de intercambio de información a través de *Internet* basado en *XML*. Este tiene la ventaja de funcionar sobre el protocolo *HTTP* que es soportado por todos los exploradores y servidores de *Internet*. Esto tiene la ventaja de evitar los siguientes problemas:

- Compatibilidad entre cliente y servidor.
- Seguridad.

SOAP (*Simple Object Access Protocol*) de alguna manera puede ser visto como una implementación de *RPC* (*Remote Procedure Call*) sobre *HTTP*.

Al funcionar a través del intercambio de documentos *XML*, está siendo ampliamente difundido como el esqueleto para una nueva generación de aplicaciones distribuidas multiplataforma y multilenguaje.

El sitio oficial es: http://www.w3.org/TR/soap/

AXIS:

Es, esencialmente un motor para *SOAP*. Un marco de trabajo que permite construir procesos que intercambien mensajes a través de *SOAP*, como clientes, servidores, etc.

La versión utilizada está escrita en *Java*. Esta versión (1.2) consta de las siguientes características:

- Velocidad: utiliza un nuevo sistema de separación e interpretación de texto que lo hacen mucho más rápido que las versiones anteriores.

- Flexibilidad: la arquitectura de *AXIS* permite al desarrollador la más amplia libertad para procesar cabeceras o modificar ciertos aspectos del funcionamiento del sistema.
- Estabilidad: se ha definido una interfaz que cambia lentamente en comparación con las anteriores versiones de *AXIS*.
- Desarrollo Orientado a Componentes: se pueden desarrollar fácilmente sistemas para ser utilizados por varias aplicaciones o para ser distribuidos.
- Soporte para *WSDL (Web Service Description Language)*:. Este permite desarrollar fácilmente *stubs* para acceder a servicios remotos. También sirve para exportar automáticamente descripciones de los servicios desarrollados con *AXIS*.

El sitio oficial es: http://ws.apache.org/axis/

Tomcat 5:

Es un contenedor de aplicaciones *Java* de tipo servidor. O sea, encapsula aplicaciones que van a interactuar con diferentes clases de clientes. Este es de utilización gratuita y con código de libre distribución (*OpenSource*).

Este consta de algunas características interesantes que lo hacen codiciado para desarrollar una aplicación similar al objetivo de este trabajo:

- Es de acceso libre.
- Es de código fuente abierto.
- Maneja aspectos de seguridad del sistema para mantener la privacidad de las aplicaciones en el servidor.
- Ofrece versiones estables y sujetas a gran cantidad de diversas pruebas.
- Está siendo utilizado por gran cantidad de empresas para ofrecer sus servicios y aplicaciones a sus clientes, lo que hace que la probabilidad de encontrar nuevos errores y problemas no detectados anteriormente sea baja.

El sitio oficial es: http://Java.sun.com/products/jsp/tomcat/

4.6.2 Tecnologías y componentes utilizados en el cliente:

A continuación se detallan las tecnologías y componentes que fueron utilizadas para la implementación del cliente de la aplicación.

Prefuse:

Es un conjunto de herramientas para crear visualizaciones altamente interactivas de conjuntos de datos estructurados y no estructurados. Esto incluye cualquier información que pueda ser representada como un conjunto de entidades posiblemente conectadas por cualquier cantidad de relaciones. Algunos ejemplos de tipos de datos soportados por *Prefuse* incluyen jerarquías (organizaciones, gráficas, taxonomías y sistemas de archivos), redes (redes de computadoras, redes sociales) y datos no estructurados (líneas de tiempo, gráficos).

Con esta herramienta, es posible realizar interfases gráficas animadas con excelente velocidad de respuesta. Además posee otras características de interés:

- Tiene un diseño orientado a objetos que permite una alta rehusabilidad de cada uno de sus componentes.
- Su arquitectura fue diseñada para el soporte de aplicaciones de gran escala.
- Ofrece soporte para agregar animaciones a las visualizaciones ofrecidas.
- Consta de una completa documentación para su correcta utilización y desarrollo de nuevas aplicaciones utilizándola como base.
- Existen componentes ya implementados para todas las interfases propuestas, lo que hace que con pocas líneas de código se tenga una visualización simple de la información deseada.

Dado que es realmente simple de usar y permite especificarle una gran cantidad de parámetros y funcionalidades, era la biblioteca deseada para desarrollar el trabajo sin perder demasiado tiempo en el desarrollo del componente de visualización.

El sitio de desarrollo de *Prefuse* es http://prefuse.sourceforge.net

AXIS:

Comentado en la sección anterior.

4.7 Diseño

En esta sección se describirá el diseño del meta-motor de búsqueda desde un punto de vista técnico. Se comienza con una breve descripción de los requerimientos que motivaron las decisiones tomadas.

El sistema requiere la utilización de un diccionario (en este caso *WordNet*) y un manejador de bases de datos (*DBMS*), pero está destinado a usuarios de baja experiencia. Como no es posible pretender que un usuario común instale un *DBMS* y un diccionario como *WordNet* en su computadora, es necesario pensar en una arquitectura cliente/servidor. La desventaja principal encontrada en este tipo de sistemas es la necesidad de una conexión con el servidor por parte de los clientes, lo que se traduce en el requerimiento de contar con acceso a *Internet* para utilizar el sistema. Sin embargo, dado que la finalidad del meta-motor es la búsqueda en la *Web*, el acceso a *Internet* se da por descontado.

El hecho de tener que dividir la aplicación en dos grandes componentes trae acarreado consigo algunas consecuencias a tener en cuenta:

- Deben dividirse funcionalidades entre ambos componentes.
- Deben definirse interfases de comunicación.
- Resulta dificultosa su verificación.
- La aplicación cliente debe ser pequeña y fácil de instalar.
- La aplicación servidor debe soportar concurrencia.

A continuación se describen los requerimientos que llevaron al diseño del servidor posteriormente descrito. La misma secuencia se sigue para especificar el diseño del cliente.

4.7.1 Requerimientos del servidor

Ante la necesidad de utilizar un diccionario para buscar términos relacionados, se extrae que el servidor debe poder interactuar con *WordNet*. Además se desea guardar la información de una ontología (la que ofrece *ODP* o una de formato similar) en una base de datos para lograr un acceso simple y evitarse problemas de concurrencia. Por esto se requiere también la utilización de un *DMBS* de uso libre (como *MySQL*).

El servidor debe poder ser implantable en diferentes arquitecturas y sistemas operativos. Esto reafirma la idea de utilizar las aplicaciones ya mencionadas debido a que estas ofrecen versiones para los sistemas más importantes de plaza.

La interacción cliente servidor debe llevarse a cabo mediante la utilización de tecnologías de uso libre no propietarias, que además permita el desarrollo de distintos tipos de clientes. Esta interacción será comentada más adelante en la sección 4.8: Arquitectura Cliente/Servidor.

4.7.2 Diseño del servidor

Los requerimientos del servidor apuntan claramente hacia la utilización de la plataforma *Java*, ya que esta cumple con los requerimientos especificados y cuenta con algunas características que ayudan en el desarrollo de la aplicación:

- Es utilizable en diferentes sistemas operativos.
- Es orientado a objetos (*Object Oriented – OO*) lo que facilita la extensibilidad y flexibilidad de la aplicación.
- Cuenta con una biblioteca de acceso a datos en manejadores de bases de datos *MySQL*.
- Cuenta con una biblioteca de acceso a la información de *WordNet*.

Además ofrece la posibilidad de utilizar algunos servicios útiles para desarrollar aplicaciones servidor, y que estas presenten un fácil acceso a diferentes clientes.

Se decidió implementar la interacción entre el cliente y el servidor mediante la utilización de *WebServices* basados en *SOAP*, ya que estos son un estándar utilizado por miles de aplicaciones. Por otro lado, los *WebServices* basados en *SOAP* pueden ser consumidos por aplicaciones escritas en varios lenguajes, lo que se traduce en un alto grado de interoperabilidad.

4.7.3 Requerimientos del cliente

Dado que la metodología requiere una fuerte interacción con el usuario, es necesario desarrollar una interfaz gráfica de alta calidad, ágil y de buen aspecto. Esta interfaz debe ser implementada en el lado cliente de la aplicación, por lo que se dispone de todo el poder de procesamiento del *PC* del usuario

Actualmente, la hegemonía del sistema operativo *Windows* como única plataforma utilizada por los usuarios comunes de computadoras de escritorio está siendo desplazada por la proliferación de sistemas operativos alternativos, como ser *Linux*, *MacOS*, *BeOS*, etc. El cliente debe ser capaz de funcionar en cualquiera de estos entornos, y no es posible (dados los cortos tiempos con los se cuenta para un proyecto de este tipo) desarrollar versiones diferentes para cada plataforma.

Es necesario utilizar tecnologías de libre distribución para que el uso de la aplicación no traiga consigo la necesidad de contratar alguna clase de licencia.

Es interesante de ser posible, obtener herramientas de código fuente libre (*OpenSource*) que tienen la importante característica de no ocultar la implementación de la aplicación al programador, y dejarlo reparar errores en caso de encontrar alguno.

4.7.4 Diseño del cliente:

El diseño del cliente es un punto crucial para el objetivo de realizar una interfaz apropiada para su utilización por manos de usuarios inexpertos. Para esto, se debió previamente decidir que tecnologías utilizar.

4.7.4.1 Tecnologías a utilizar

Dados los requerimientos del cliente, las opciones manejadas fueron las siguientes:

Realizar un cliente totalmente *Web*, basado en tecnologías como *Javascript* y soporte desde el servidor con lenguajes para desarrollo en un entorno *web* como JSP, *ASP*, *PHP*, *Perl* u otro similar.

Este tipo de aplicaciones tiene las siguientes ventajas:

- No es necesario instalar una aplicación en el *PC* de los usuarios
- Es posible obtener un producto que funcione con varios navegadores, lo que soluciona el requerimiento de funcionar en varias plataformas.

Sin embargo, también presenta las siguientes desventajas:

- El hecho de usar *JSP*, *ASP* o similares hace que parte del procesamiento de la presentación recaiga sobre el servidor, aumentado considerablemente la dificultad de escalar el sistema.
- El costo de desarrollo de este tipo de aplicaciones es elevado.
- Desarrollar una interfaz gráfica como la que se desea presenta una complejidad fuera de alcance.
- Lograr que la aplicación funcione correctamente en varios navegadores indica también una dificultad adicional no despreciable.
- No hay que realizar actualizaciones del lado del cliente.

Otra opción manejada, fue realizar un cliente *web* basado en *Applets Java*. Los *Applets* son programas escritos en *Java* que tienen la interesante característica de poder ser incluidos en una página *web* de la misma manera que es incluida una imagen en estos sitios. Cuando se navega en una página que contiene un *Applet*, el código de este es transferido al sistema y ejecutado por la *JVM* que contiene el navegador utilizado.

Ventajas:

- La mayoría de los navegadores soporta la utilización de *Applets Java*.
- El desarrollo de la interfaz gráfica es mucho más simple que en la opción anterior.
- *Java* es multiplataforma lo que hace al cliente utilizable en diferentes sistemas operativos.
- No es necesario instalar una aplicación en el cliente. Y como consecuencia de esto, nunca se deben realizar actualizaciones del lado del cliente.

Desventajas:

- El *Applet* debe ser descargado cada vez que el usuario desee utilizar la aplicación.
- Es necesario instalar la *JRE* (*Java Runtime Environment*).
- Es común que los *Java Applets* no funcionen en ciertas computadoras debido a problemas de configuración de la *JRE*.
- No es posible (al menos de forma elegante y sencilla) mantener información en el *PC* del cliente.

Evaluando ventajas y desventajas, se tomó la decisión de desarrollar una aplicación de escritorio en lenguaje *Java* que puede ser utilizado sin problemas en cualquiera de los sistemas operativos mencionados anteriormente. Esto se debe a que todos ellos tienen disponible una versión de la *JVM* (*Java Virtual Machine*), que se encarga de interpretar el código implementado.

Ventajas:

- Permite gran flexibilidad para desarrollar la interfaz gráfica.
- Posibilita mantener información en el cliente.
- Mantiene todo el procesamiento de interacción con el usuario en el lado del cliente.
- El costo de desarrollo es menor a la opción enteramente *web*.

Desventajas:

- El usuario debe descargar e instalar una aplicación.
- El usuario debe instalar la *JRE*.
- Si se libera una nueva versión del cliente, ésta deber ser descargada por los usuarios, lo que se maneja automáticamente en las opciones anteriores.

Los factores que más influyeron en la decisión del tipo de cliente a desarrollar fueron los cortos tiempos de desarrollo y la necesidad de obtener una interfaz de usuario de alta calidad. Sin embargo, dada la implementación del servidor, es fácil desarrollar otro tipo de clientes.

Entre las opciones posibles se puede encontrar:

- Migrar la aplicación de escritorio a un *Applet* (escasa complejidad)
- Realizar un cliente *web* basado en el conjunto de Tecnologías *AJAX*.
- Implementar un cliente utilizando la plataforma *.NET*.

En términos generales, cualquier plataforma capaz de manejar la tecnología *SOAP* puede ser utilizada para desarrollar un cliente.

4.7.4.2 Diseño de la interfaz

Como ya se comentó, la interfaz es un punto crucial en la utilidad que pueda tener luego la aplicación. Al ser este un sistema de ayuda a usuarios inexpertos, está claro que debe resultar intuitiva y por lo tanto fácil de utilizar.

La diagramación de la interfaz se dividió en dos ventanas centrales: una ventana de visualización, donde se muestra la información relacionada con los resultados de la ontología, y una de búsqueda que ofrece la posibilidad de ingresar los datos iniciales.

Además, existía la necesidad de generar de alguna manera un perfil de usuario, por lo cual también existe una ventana destinada al manejo de esta exigencia.

La aplicación consta de algunas opciones de configuración que son manejadas en la ventana de preferencias.

La ayuda para utilizar la aplicación cliente es preponderante para los usuarios inexpertos. Por tanto se decidió generar dos sistemas de ayuda:

- Ayuda en línea: que se presenta como una ventana aparte que va mostrando la serie de pasos a seguir para generar una consulta.
- Ayuda extendida: se muestra a través de una ventana del navegador que posea el sistema. Contiene información detallada de cada una de las utilidades de la aplicación y muestra imágenes explicativas de las ventanas de esta. Está generada en *HTML* para su fácil acceso.

Panel de Búsqueda:

El panel de búsqueda permite el ingreso de la consulta inicial. Una vez obtenidos los resultados, estos se muestran en el panel de visualización. Cada vez que se elige una categoría del árbol de resultados, esta aparece en la lista de categorías de este panel. Enseguida se piden sugerencias para los términos ingresados inicialmente que van a aparecer en la lista de sugerencias. Aquí se pueden elegir términos relacionados para expandir la consulta. Una vez terminado este proceso, se puede elegir el buscador a utilizar y se presiona el botón para generar la consulta.

El diseño de este panel se basa en la secuencia de actividades que debe realizarse para generar una consulta correctamente. De esta manera están dispuestas las alternativas para realizar el trabajo comenzando con el ingreso de las palabras en la parte más alta de la barra de búsqueda, y terminando con la generación de la consulta en la parte más baja.

La Figura 4.34 muestra la estructura de este panel, conteniendo de arriba hacia abajo:

- Campo de ingreso de consulta.
- Cuadro con categorías seleccionadas.
- Botón para pedir sugerencias.
- Cuadro con sugerencias para cada término ingresado.
- Botón para generar la consulta.

Figura 4.34 Panél de Búsqueda

La secuencia a realizar para obtener una consulta es la siguiente:

- Ingresar palabras.
- Apretar botón de búsqueda.
- Elegir temas relacionados en el Panel de Visualización de los resultados de la ontología.
- Pedir sugerencias.
- Navegar sobre la lista de sugerencias y marcar las que se consideren útiles.
- Apretar el botón de generación de la consulta con el buscador deseado.

Esta secuencia de pasos permite generar una búsqueda correctamente. Como se puede observar, la interfaz está diseñada para que esta serie de pasos resulte intuitiva. Sin embargo, para iniciar a los usuarios en la forma de utilización del sistema se agregó una ventana de ayuda que se comentará en el ítem: Ventana de ayuda en línea (dentro de esta misma sección).

Existe la posibilidad de enviar la consulta a otro buscador, para esto se debe presionar el botón derecho del ratón sobre el botón con la etiqueta "*Google*" y elegir el buscador deseado. En esta versión de la aplicación cliente están disponibles tres buscadores:

- *Google*
- *Yahoo*
- *Altavista*

En las figuras Figura 4.35 y Figura 4.36 se pueden ver los botones que aparecen para buscar en otros sitios.

Figura 4.35 Buscar utilizando *Yahoo*

Figura 4.36 Buscar utilizando *Altavista*

Panel de Visualización:

El panel de visualización se centra en la capacidad de mostrar los resultados obtenidos de la ontología, a partir de las palabras ingresadas inicialmente.

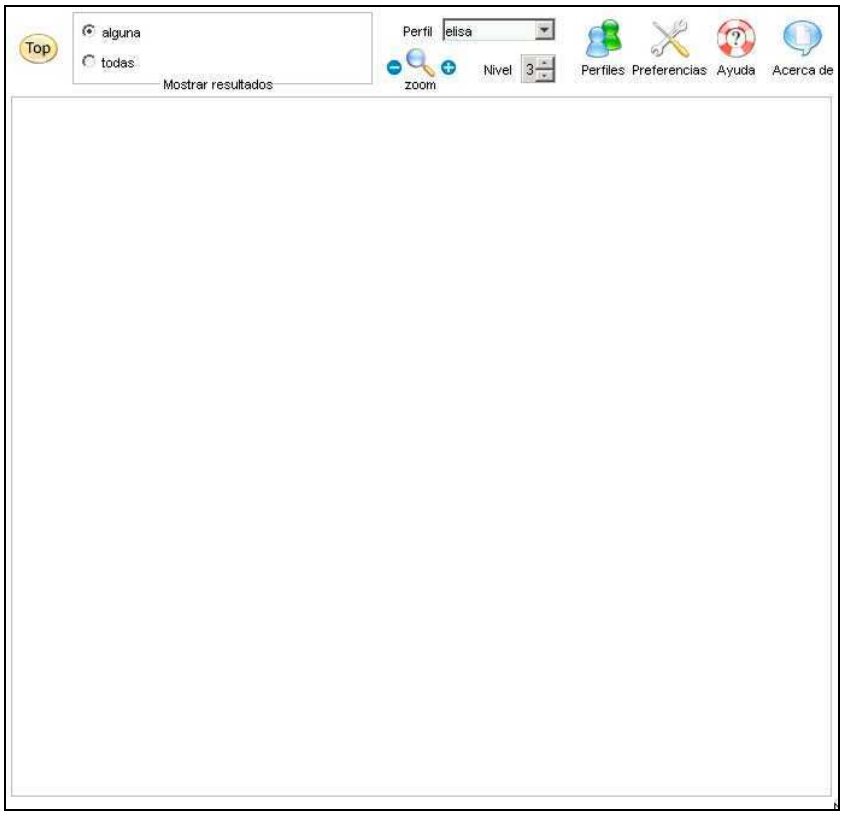

Figura 4.37 Panel de visualización

Como se puede observar en la Figura 4.37, este cuenta con una gran cantidad de opciones de visualización que se pasan a comentar y explicar a continuación.

Los resultados son dispuestos en forma jerárquica para obtener conocimiento de las relaciones entre las categorías.

En ciertas ocasiones sucede que es retornada una gran cantidad de resultados, lo cual dificulta las posibilidades de visualización y búsqueda de información relacionada. Para esto, se diseñaron algunas opciones que permiten mejorar la visualización de los datos. Estas opciones son:

- La posibilidad de volver a la raíz de la jerarquía (botón con etiqueta "*Top*").
- Controles de filtrado.
- Controles de *zoom*.
- Selector de nivel.

Cuando se está navegando sobre la jerarquía de resultados suele ser cómodo retornar rápidamente a la raíz para tomar otro camino y continuar buscando información relacionada. Para esto se agregó el Botón para ir a la raíz de los resultados.

Los controles de filtrado permiten elegir los resultados relacionados con algunas de las palabras ingresadas en el panel de búsqueda. En este aparecen todas las palabras escritas y uno puede elegir mostrar los resultados que se relacionen con algunas de ellas. También se puede cambiar el modo de despliegue de la información y obtener los resultados que están relacionados con todas las palabras ingresadas (o un subconjunto de ellas). Estas opciones permiten filtrar los resultados para poder observarlos más cómodamente.

En caso de no desear filtrar los resultados y querer observarlos todos, se puede acercar la imagen para poder observarlos más claramente. Si la situación es opuesta, se puede alejar la imagen y ver los resultados en forma global.

El control de nivel permite observar más o menos niveles en la jerarquía de resultados. De esta manera es posible obtener resultados de distintos niveles sin la necesidad de navegar por la jerarquía.

Ventana de perfil de usuario:

Esta ventana es la encargada de interactuar con el usuario para realizar todas las tareas relacionadas con la generación, edición y destrucción de los perfiles de usuario.

Figura 4.38 Perfil de usuario

Como se puede observar en la Figura 4.38 esta consta de un panel central donde se puede observar la jerarquía de términos a seleccionar para generar un nuevo perfil, o editar uno ya existente.

En caso de querer generar uno nuevo, se debe previamente apretar el botón con la etiqueta "Nuevo" e insertar el nombre con el cual se desea que este perfil perdure. Si en cambio se quiere editar un perfil ya existente se debe marcar previamente el deseado del selector de perfiles.

Una vez realizado uno de los pasos anteriores, se pueden marcar las categorías pretendidas de la jerarquía ofrecida. Se puede además navegar sobre la jerarquía un nivel más del mostrado, esto se debe a la posible necesidad de agregar alguna categoría más específica. Una vez seleccionadas las categorías se pasa a guardar el perfil presionando el botón "Guardar".

En caso de querer borrar un perfil se lo elige dentro del selector de perfiles y se presiona el botón de "Eliminar".

Como se puede observar, los botones están ordenados de izquierda a derecha con los pasos a dar para generar, editar o borrar un perfil.

En la zona inferior de la pantalla se agregó el botón destinado a volver a la aplicación para comenzar a trabajar.

Ventana de preferencias:

Desde esta ventana se pueden configurar las opciones principales de la aplicación. Para su mejor utilización y comprensión, fue diseñada como un conjunto de hojas distintas, separando en cada una las diferentes opciones. Esta consta de 3 hojas:

- Configuración del Servidor (Figura 4.39).
- Selección de Idioma (Figura 4.40).
- Configuración del *Proxy* (Figura 4.41).

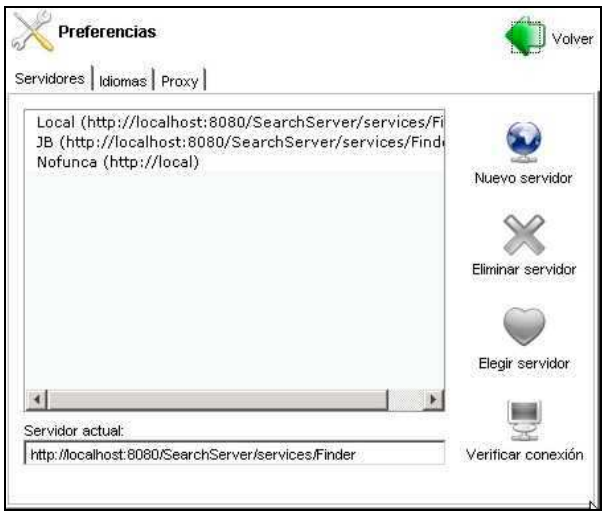

Figura 4.39 Preferencias: Servidores

La hoja de configuración del servidor permite elegir a cual servidor se desea que la aplicación se conecte. Uno puede cambiar de servidor cuando se está utilizando la aplicación en caso de que el utilizado hasta el momento haya dejado de funcionar o esté retornando resultados muy lentamente.

Las opciones de las que se dispone son:

- Agregar un nuevo servidor a la lista.
- Elegir uno diferente.

- Eliminar un servidor de la lista.
- Verificar si uno está disponible para ser utilizado.

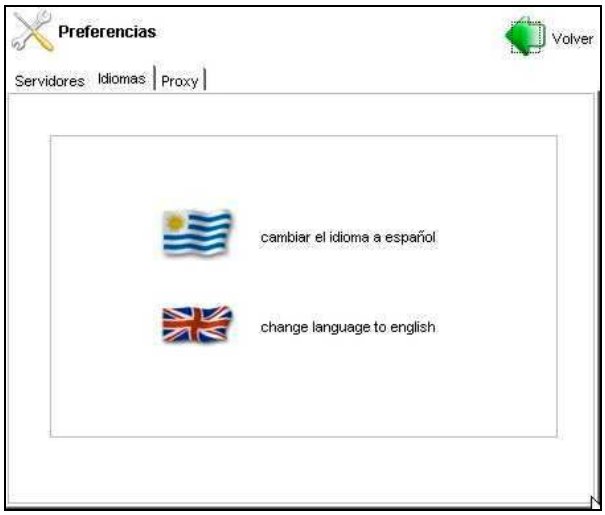

Figura 4.40 Preferencias: Idiomas

En la hoja de selección de idioma se puede elegir el idioma en que la interfaz va a estar escrita. Actualmente se puede elegir entre el Inglés y el Español, pero existe la manera de agregar fácilmente un nuevo idioma a la aplicación generando un archivo con las palabras a mostrar en la interfaz. El idioma de la aplicación puede ser cambiado en cualquier momento y sin necesidad de reiniciarla.

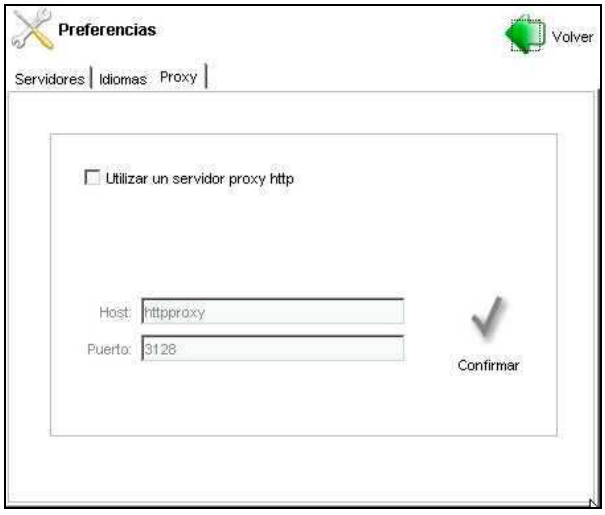

Figura 4.41 Preferencias: Selección de *Proxy*

En la sección de Configuración del *Proxy* se puede especificar la información para utilizar un *Proxy* cuando se requiere conectarse al servidor.

Esta ventana también tiene la opción de retornar a la aplicación presionando el botón "Volver".

Ventana de ayuda en línea:

Esta proporciona la serie de pasos a seguir para utilizar fácilmente la aplicación. Está diseñada como una ventana aparte de las dos centrales que aparecen cuando se inicia el sistema. Consta de un cuadro con texto donde se explica el próximo paso a dar para generar una consulta.

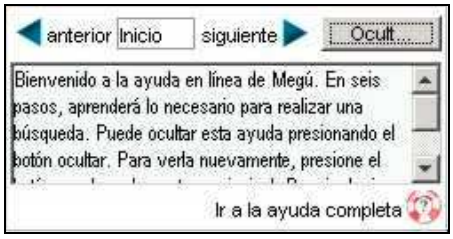

Figura 4.42 Ayuda en línea

La ventana tiene la opción de poder ser ocultada en caso de que el usuario se considere lo suficientemente ágil en el sistema como para no utilizarla.

Ayuda extendida:

Esta ayuda está escrita en *HTML* para su fácil acceso y lectura, y proporciona una explicación detallada de cada una de las alternativas que ofrece la aplicación. Consta de un documento específico para la explicación de las funcionalidades de cada ventana. Además se puede seleccionar (hacer *click* con el ratón) sobre la información de las imágenes mostradas para obtener comentarios al respecto.

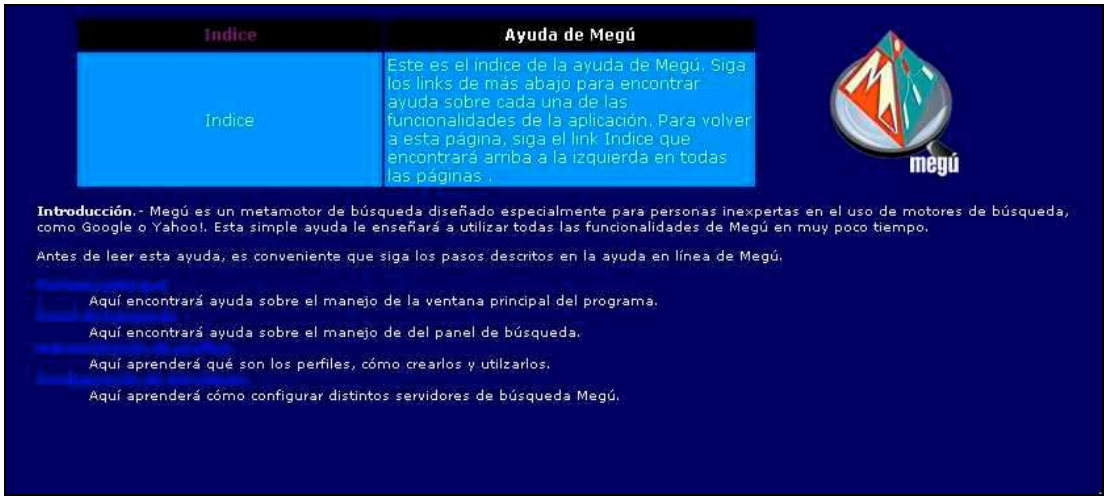

Figura 4.43 Ayuda extendida

4.8 Arquitectura Cliente – Servidor

La interacción entre el cliente y el servidor está implementada mediante el intercambio de mensajes *SOAP* (ver sección 4.6 – Tecnologías Utilizadas). La idea principal es que la aplicación cliente realice consultas sincrónicas sobre el servidor, obtenga los resultados, los visualice correctamente y soporte la interacción con el usuario. Dicha interacción podrá generar nuevas consultas con su posterior despliegue de resultados.

Es importante que la arquitectura desarrollada ofrezca características como:

- No presentar problemas de seguridad ni al servidor ni a los clientes, o por lo menos no incrementarla.
- Permitir el desarrollo de nuevos clientes que utilicen el mismo servidor.
- Tener una interfaz clara de comunicación entre ambas partes.
- Minimizar el tiempo de espera de la aplicación cliente.

A continuación se detallan las características de ambos componentes (cliente y servidor).

4.8.1 Servidor

Las principales metas que debe cumplir un servidor de este tipo son:

- Soportar la interacción con varias aplicaciones cliente al mismo tiempo (concurrencia).
- Ser eficiente en el procesamiento de las consultas (de forma de minimizar los tiempos de respuesta)
- Ser eficiente en la utilización del ancho de banda.
- Ser fácilmente escalable.

Cada una de las metas descritas fue tomada en cuenta en el diseño del sistema creado. Algunas de las metas plantearon problemas de difícil solución, por ejemplo la escalabilidad del sistema con respecto a la cantidad de clientes que se conectan a un mismo servidor. En casos como el anterior y debido a las restricciones de tiempo, se decidió mantener el diseño ampliable, simple y abierto (de forma de que la solución pueda ser implementada con poco trabajo en un futuro), en lugar de intentar resolver a medias el problema.

Para cumplir con las metas planteadas, se tomaron las siguientes decisiones:

- El formato para el intercambio de mensajes es simple. Esto permite que se realicen distintas implementaciones del servidor o del cliente, manteniendo la interoperabilidad.
- El servidor no mantiene estados ni sesiones. De esta forma, cada consulta es independiente de las anteriores, lo que simplifica enormemente la escalabilidad y el manejo de concurrencia. Por otro lado también simplifica la implementación de nuevos servidores. Sin embargo, presenta la propiedad no deseada de tener que retrasmitir cierta información cada vez que es enviado un mensaje al servidor.
- El acceso a la base de datos de categorías es de solo lectura. Esto permite acceder a la base de datos de forma no transaccional (ya que no se presentan mayores problemas de concurrencia), lo que mejora los tiempos de acceso y obtención de resultados.

El principal problema de la interacción entre el cliente y el servidor, es la gran cantidad de datos que se transmiten como respuesta a algunas consultas (aquellas que dan muchos resultados). Este tráfico puede ocasionar que el tiempo de respuesta a una consulta sea demasiado largo (dependiendo del ancho de banda disponible en el cliente y en el servidor).

Para mejorar esta situación se tomaron varias medidas:

- Crear una representación de datos que reduzca el tráfico de información.
- No retornar los resultados como una colección.
- Limitar la cantidad máxima de términos de una consulta.
- Analizar la base de categorías para identificar los términos que retornan demasiados resultados y filtrar los resultados no relevantes.
- Comprimir los mensajes *SOAP* al vuelo (y descomprimirlos en el cliente).

4.8.2 Cliente

La aplicación cliente puede ser configurada para consultar distintos servidores. Esto permite que el sistema siga en funcionamiento ante la eventual caída de un servidor.

La siguiente es una descripción de una sesión típica observada desde el punto de vista de la aplicación cliente:

- El usuario escribe una consulta en la aplicación cliente.
- La consulta es emitida al servidor, y este responde sincrónicamente (el cliente queda bloqueado hasta recibir la respuesta) con el conjunto de resultados para la consulta.
- El cliente procesa la respuesta y visualiza el árbol de categorías resultante.
- El usuario marca un conjunto de categorías en el árbol y solicita sugerencias.

- La solicitud es emitida al servidor, y este responde sincrónicamente con un conjunto de sugerencias.
- Las sugerencias son procesadas y visualizadas por el cliente.
- El usuario selecciona términos del conjunto de sugerencias y cuando está satisfecho solicita la emisión de su consulta a un motor de búsqueda (por ejemplo: *Google*). Para esto, sucede que se envía un mensaje al servidor conteniendo toda la información relevante de la consulta. Este responde con una consulta en el formato adecuado para ser utilizada en un motor de búsqueda específico. Finalmente la aplicación cliente emite dicha consulta el motor de búsqueda.

Es importante destacar que es la aplicación cliente la que interactúa con el motor de búsqueda. Esto se da de esta manera por las siguientes razones:

- Para protegerse de posibles abusos, los motores de búsqueda no permiten que se realicen más de un determinado número de consultas desde una misma dirección *IP* por período de tiempo.
- La interacción con el motor por parte del servidor sobrecargaría fuertemente la conexión al incrementar enormemente el uso del ancho de banda.

4.8.3 Mensajes cliente/servidor

El intercambio de mensajes entre el cliente y el servidor es siempre iniciado por el cliente realizando consultas. El servidor responde estas consultas sincrónicamente con los resultados obtenidos.

A continuación se especifica la manera en que se generan y reciben los mensajes entre el servidor y el cliente para las funcionalidades más importantes que ofrece el servidor. Si se desea obtener más información sobre estas funcionalidades y las restantes que el servidor ofrece, ver el Apéndice A: Algoritmos.

Consulta de categorías:

El primer paso consiste siempre en la formulación por parte del usuario de una consulta. Dicha consulta está constituida por un conjunto de términos separados por espacios. Luego de formulada, esta es enviada al servidor. El servidor consulta la base de categorías (mediante el uso de un índice invertido) y obtiene el árbol de resultados. En ocasiones, el árbol de resultados puede ser muy grande, lo que deteriora los tiempos de respuesta si el ancho de banda disponible es escaso. Debido a esto, el servidor procesa los resultados y los representa de manera que ocupen menos espacio. Los resultados son retornados como un conjunto de nodos que conforman un árbol. Cada nodo tiene una etiqueta (*Art*, *Movies*, etc…), un identificador (*Id*) y el identificador de su padre (*IdP*). La raíz es reconocida debido a que el *IdP* coincide con el suyo propio.

La representación (simplificada) contiene los siguientes datos:

- Etiqueta de la categoría.
- *Id.*
- *IdP*.

Sin embargo, esta reducción no es suficiente para lograr una tasa de transmisión de datos satisfactoria. Supóngase que el conjunto de resultados consta de 5000 categorías, la reapuesta del servidor consistirá de una colección de 5000 ternas de la forma descrita arriba. Como el intercambio de mensajes se hace transmitiendo documentos *XML* (mediante el protocolo *SOAP*), es imperativo unir todos los resultados en un *String* ya que, de otra manera, se agregaría un par de etiquetas *XML* para cada elemento del conjunto de datos a transmitir.

El siguiente ejemplo explica la situación:

Suponga que se desean transferir las categorías:

- Top

- Top/Arts
- Top/Business
- Top/Arts/Movies

Inicialmente se generaría el siguiente documento *XML* a enviar:

```
<secuencia>
  <elemento>Top</elemento>
  <elemento>Top/Arts</elemento>
  <elemento>Top/Business</elemento>
  <elemento>Top/Arts/Movies</elemento>
</secuencia>
```

Figura 4.44 Formato inicial del documento *XML* a transmitir

Aplicando la traducción especificada más arriba, el conjunto de datos se transforma en:

- Top-0-0
- Art-1-0
- Business-2-0
- Movies-3-1

Entonces el documento *XML* que se transmitirá será algo similar a:

```
<secuencia>
  <elemento>Top-0-0</elemento>
  <elemento>Art-1-0</elemento>
  <elemento>Business-2-0</elemento>
  <elemento>Movies-3-1</elemento>
</secuencia>
```

Figura 4.45 Formato del documento *XML* a transmitir antes su compresión

Como se puede observar, es posible eliminar las etiquetas *XML* uniendo todos los datos en una única cadena de caracteres. Al unir estos, el documento *XML* será similar a:

```
<secuencia>
  <String>Top-0-0*Art-1-0*Business-2-0*Movies-3-1</String>
</secuencia>
```

Figura 4.46 Formato del documento *XML* a transmitir luego de la compresión

Nota: (El * fue utilizado como carácter separador en este ejemplo)

Como es fácilmente apreciable, las etiquetas <*elemento*></*elemento*> introducen una enorme sobrecarga innecesaria a la hora de transferir los resultados. Aunque unir los resultados en un solo *String* es menos elegante, los tiempos de transferencia se reducen en promedio en un 70 %.

Consulta de sugerencias:

Luego de haber obtenido y revisado el árbol de resultados, el usuario marca categorías que le parecen relevantes. Las categorías que selecciona se utilizan para desambiguar el significado de los términos de la consulta inicial y sugerir nuevos términos (sinónimos e hiperónimos extraídos de *WordNet*). La obtención de sugerencias se lleva a cabo de la siguiente manera:

La aplicación cliente envía al servidor:

- Términos de la consulta inicial
- Categorías marcadas en el árbol.

Notar que podría haberse obviado el hecho de mandar los términos de la consulta inicial si se manejara el concepto de sesión, pero esto requeriría aumentar considerablemente la complejidad del servidor. Dicho aumento de complejidad no es justificable bajo ningún concepto, ya que la cantidad de información redundante que se transmite es despreciable (la consulta del usuario es un pequeño conjunto de palabras, insignificante cuando se lo compara con los cabezales de un mensaje *SOAP*).

La respuesta del servidor al pedido de sugerencias es, para cada palabra de la consulta inicial, un conjunto de términos relacionados. En este caso no existen problemas de ancho de banda, ya que la cantidad de información transmitida es muy pequeña.

Generación de la consulta para el motor de búsqueda:

El paso final consiste en la generación de la consulta que efectivamente se emitirá al motor de búsqueda que prefiera el usuario.

Para esto, se envía un mensaje al servidor conteniendo:

- Términos de la consulta inicial.
- Categorías seleccionadas del árbol.
- Sugerencias elegidas en el paso anterior.

El servidor responde entonces con una consulta en el formato apropiado para ser enviada a un motor de búsqueda. La utilización del ancho de banda tampoco es un problema en este paso.

A continuación se muestra un diagrama que resume la interacción entre el cliente y el servidor:

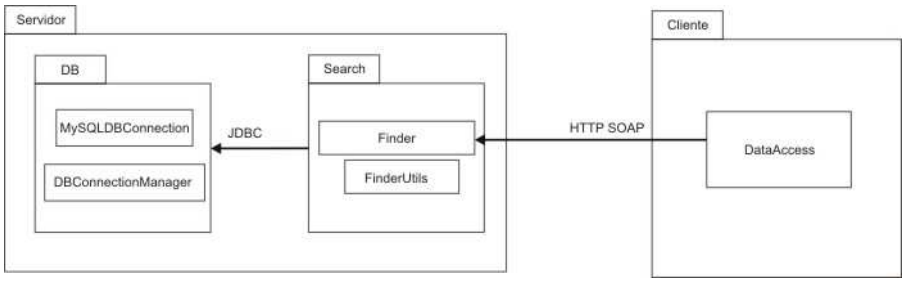

Figura 4.47 Interacción cliente - servidor

4.9 Implementación

Esta sección se refiere a la implementación de la aplicación. Se comenzará con el servidor de búsquedas para luego pasar al cliente.

4.9.1 Implementación del servidor

El servidor de búsqueda fue implementado completamente utilizando el lenguaje *Java* (versión 1.4.2 de la *J2SE*). Como se describió en la sección de diseño, el servidor es visto desde el punto de vista del cliente como un conjunto de métodos consumibles mediante el uso de *WebServices*.

Dado que los *WebServices* son una tecnología relativamente nueva, se pasa ahora a describir sucintamente el concepto de *WebServices*. La descripción no intenta ser rigurosa y mucho menos completa; su objetivo es permitir a aquellos que no conozcan esta tecnología poder entender esta sección del documento.

¿Qué significa WebService?

Un W*ebService* es una colección de protocolos y estándares utilizados para intercambiar datos entre aplicaciones. Las aplicaciones de software escritas en diversos lenguajes y para diversas plataformas pueden utilizar esta tecnología para intercambiar datos sobre redes de computadoras como *Internet*, de un modo similar a la comunicación entre procesos en una sola computadora. Esta capacidad de interoperabilidad se debe principalmente al uso de estándares abiertos.

¿Cómo funcionan?

Una descripción muy primitiva y simplificada podría ser como la que se da a continuación.

Una aplicación corriendo en una computadora A desea cierta obtener información. En una computadora B existe una aplicación que es capaz de dar dicha información. Las dos computadoras están conectadas mediante una red y la computadora B publica los servicios que provee implementando un *WebService*. La computadora A envía un mensaje de texto, basado en *XML*, a la computadora B. El mensaje de texto viaja sobre el protocolo *HTTP*, de la misma forma que cuando un navegador de *Internet* (*browser*) solicita a un servidor *web* una página de *Internet*. La computadora B recibe el mensaje, interpreta el documento *XML* y genera una respuesta, también descrita en *XML* que es transmitida a la computadora A. Lo que permite la interoperabilidad es el hecho de que los mensajes están especificados en *XML*. No hay referencias al lenguaje o la plataforma en que las dos aplicaciones están escritas. No hay objetos "binarios" codificados en los mensajes.

Ventajas de los *WebServices*:

- Proveen interoperabilidad entre varias aplicaciones corriendo en plataformas dispares.
- Los *WebServices* usan estándares y protocolos abiertos. Los protocolos y formatos de intercambio de datos están basados en texto, permitiendo a los desarrolladores comprenderlos.
- Al utilizar el protocolo *HTTP*, los *WebServices* pueden funcionar a través de *firewalls* y otras medidas de seguridad sin necesidad de realizar cambios. Por ejemplo: cambiar las reglas de filtrado.
- Permiten la reutilización de servicios y componentes en una infraestructura.

Desventajas de los *WebServices*:

- Los *WebServices* no ofrecen manejo de transacciones (por lo menos no como el ofrecido por otros estándares más maduros como *CORBA*).
- Sufren ciertos problemas de desempeño (*performance*) cuando se los compara con otras aproximaciones como *RMI* (*Remote Method Interface*) o *CORBA*. Esto ocurre debido al uso de formatos basados en texto. La eficiencia no es un objetivo explícito de *XML*.

¿Cómo se consume un *WebService* desde un cliente?

Cuando una aplicación publica ciertos servicios en forma de *WebService*, se genera un documento basado en *XML* llamado vulgarmente "el *WSDL*" (*Web Services Description Language*). Dicho documento contiene una detallada descripción de los servicios publicados. Esta descripción especifica los argumentos que deben ser pasados para invocar un servicio, el nombre del servicio y el formato en que la respuesta a un pedido de servicio será especificada. El *WSDL* debe ser visible para el cliente, y a partir del mismo es posible implementar los mecanismos de invocación de los servicios. Aunque es posible que un cliente genere los mecanismos de invocación en tiempo de ejecución (esto es luego de que el cliente fue programado, o sea, cuando está siendo ejecutado), lo más común es que el programador del cliente utilice el *WSDL* para obtener los mecanismos de invocación necesarios para consumir los servicios publicados

por el servidor. En *Java* (y en muchos otros lenguajes) existen utilidades que permiten generar automáticamente el código necesario para consumir un *WebService* a partir del *WSDL*, de forma prácticamente transparente para el programador.

Mapeo de tipos en *WebServices*:

Dado que los *WebServices* son independientes del lenguaje en que se implementan el servidor y el cliente, es necesario establecer un mapeo de tipos de datos. Para esto, se define un conjunto de tipos de datos comunes que incluyen *Strings* (cadenas de caracteres), enteros, números de punto flotante, etc. Los tipos de datos del lenguaje utilizado en el servidor y el cliente son mapeados a estos tipos comunes. Es incluso posible mapear tipos complejos, como colecciones *Java*. Para esto, la especificación de *WebServices* incluye definiciones de tipos secuencia, arreglo, etc.

4.9.1.1 Servicios publicados por el servidor de búsquedas

Se define ahora el conjunto de servicios publicados por el servidor de búsqueda y se detalla su funcionamiento.

El servidor de búsqueda encapsula todos los servicios publicados en un objeto fachada (*facade*) llamado *Finder*. Los métodos públicos del objeto *Finder* son entonces los servicios que pueden ser invocados por los clientes.

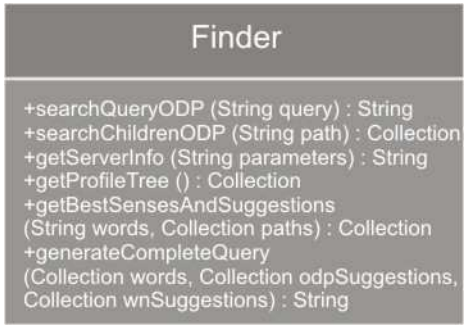

Figura 4.48 Diagrama de clase *Finder*

- `searchQueryODP (String query) : String`

Este método toma coma parámetro un *String* conteniendo palabras separadas por espacios (consulta). El cometido de este método es generar un conjunto de categorías relacionadas con las palabras de la consulta organizadas en un árbol jerárquico. Dicho árbol tiene como hojas el conjunto de categorías que se relacionan directamente con la consulta. Los nodos internos son los ancestros de dichas categorías en la ontología que se encuentra cargada en la base de datos del servidor.

La razón por la que se retorna un *String* en lugar de una colección es que la representación en lenguaje *XML* de una colección introduce una enorme carga al momento de la transferencia del resultado por una conexión de red de bajo ancho de banda (notar que la cantidad de resultados para esta operación puede ser enorme). El árbol de categorías está codificado en el *String* retornado. El formato de la codificación se encuentra especificado en el apéndice A de este documento.

- `searchChildrenODP (String path) : Collection`

Este método toma coma parámetro un *String* conteniendo un camino completo de la ontología, utilizando como separador de categorías el carácter "/". El objetivo de este método es retornar el conjunto de categorías hijas (más específicas) del camino detallado (todos los nodos del árbol de

categorías que son hijos de la última categoría del camino). Se retorna una colección de *Strings*, los cuales representan estas categorías.

- `getServerInfo (String parameters) : String`

Este método recibe un *String* conteniendo diferentes parámetros. El objetivo de esta operación es proveer de una interfaz genérica que permita al cliente obtener información del servidor (como por ejemplo el idioma, la versión, etc.). En la implementación realizada, solo se utiliza esta operación para probar la conexión con un servidor desde el cliente.

- `getProfileTree () : Collection`

Este método retorna una colección de *Strings*. Cada uno de los *Strings* de la colección representa un camino en un árbol jerárquico de categorías. El separador de categorías en los caminos es el carácter "/".

El objetivo de esta operación es que el servidor le proporcione al cliente un árbol para que este último permita al usuario definir un perfil de usuario. En la implementación realizada, esta operación retorna un árbol formado por los primeros dos niveles de la ontología contenida en la base de datos del servidor.

- `getBestSensesAndSuggestions (String words, Collection paths) : Collection`

Este método toma coma parámetro un *String* (*words*) conteniendo un conjunto de palabras separadas por espacios y una colección de caminos del árbol jerárquico de categorías. Las palabras son las especificadas inicialmente por el usuario y la colección de caminos esta dada por las selecciones del usuario en el árbol de categorías.

El objetivo de esta operación es, para cada palabra de las contenidas en el *String words*, retornar una definición de diccionario y un conjunto de palabras relacionadas (sugerencias). Puede ocurrir que para alguna de las palabras de *words* no se obtenga definición y palabras relacionadas. La colección de caminos que es tomada como parámetro cumple la función de proveer de contexto para desambiguar los términos de *words*. La especificación completa del algoritmo utilizado para implementar este método se encuentra en el apéndice A.

- `generateCompleteQuery (Collection words, Collection odpSuggestions, Collection wnSuggestions) : String`

Este método toma como parámetros una colección de palabras (*words*), una colección de caminos en el árbol jerárquico y una colección que, para cada palabra de *words*, contiene una subcolección de palabras relacionadas. El objetivo de esta operación es generar una consulta sobre un motor de búsqueda que soporte operaciones booleanas. La especificación completa del algoritmo utilizado para implementar este método se encuentra en el apéndice A.

Instalación del servidor como un *WebService*:

Como se dijo anteriormente, el servidor de búsqueda está diseñado para ser instalado como un *WebService* en un servidor de aplicaciones. Durante el desarrollo del servidor, se utilizó *Apache Tomcat* en las versiones 4 y 5. No se realizaron pruebas en otros servidores de aplicaciones, aunque no deberían existir problemas (no se utilizó ninguna funcionalidad específica de *Tomcat*).

4.9.1.2 Subsistemas

La implementación del servidor puede ser dividida en subsistemas. Aunque se realizaron esfuerzos por mantener la modularidad del código, aún hay mucho espacio para mejorar este aspecto.

A continuación se detallan cada uno de los subsistemas de la implementación del servidor y las funcionalidades que proveen.

Subsistema de acceso a la base de datos:

Este subsistema se encarga de abstraer el acceso a la base de datos. En la implementación realizada se utilizó el *DBMS MySQL* versión 4.1, y la biblioteca de *Java MySQL Connector* para acceder a *MySQL*. Reimplementando este subsistema, es posible utilizar otro *DBMS*.

El subsistema de acceso a la base de datos se encuentra encapsulado en el paquete *db*. La clase principal perteneciente a este paquete es *DBConnectionManager*. Esta clase se encarga de instanciar un controlador de conexiones a la base de datos que implemente la interfaz *IDBConnection*. Implementando esta interfaz, es posible crear un controlador para cualquier base de datos accesible vía *JDBC*. El nombre de la clase implementada para utilizar *MySQL* es MySQLDBConnection. La clase instanciada por *DBConnectionManager* es configurable (ver Apéndice C: Configuración del servidor).

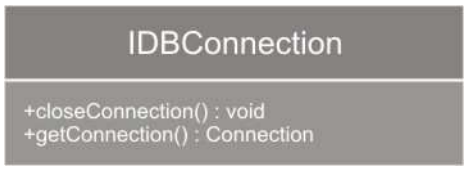

Figura 4.49 Diagrama de interfaz *IDBConnection*

DBConnection está implementada como una clase *Singleton* (solo puede existir una instancia de la misma). Esta clase implementa la interfaz *IDBConnection*. La utilización de otro *DBMS* puede llevarse a cabo implementando una nueva clase que implemente *IDBConnection*.

Subsistema de configuración del servidor:

El paquete *config* se encarga de la inicialización del servidor. Este lee la configuración del servidor desde un archivo. La ubicación de dicho archivo de configuración debe ser especificada en el descriptor del *WebService* (archivo "*web.xml*").

La clase principal de este subsistema es *ServerConfigManager*. Esta clase estática permite obtener los parámetros de inicialización a partir del archivo de configuración (ver apéndice C). Pueden agregarse nuevos parámetros al archivo de configuración sin modificar el subsistema.

Subsistema de búsqueda:

El subsistema de búsqueda está implementado en el paquete *search*. Las principales clases son *Finder y FinderUtils*. Aunque el diseño del servidor permite utilizar cualquier ontología y DBMS, el diccionario (*WordNet*) no puede ser sustituido sin un considerable esfuerzo. Esto se debe a las siguientes razones:
La *API* de *WordNet* es compleja y los algoritmos implementados dependen fuertemente de la misma.
No existen productos de uso libre similares a *WordNet*.

De todas formas, el acceso a *WordNet* está localizado en la clase *FinderUtils*. Esta clase implementa los métodos de búsqueda de sugerencias descritos en el apéndice A. La clase *Finder*, además de ser la fachada del *WebService*, implementa los algoritmos de búsqueda de categorías.

4.9.2 Implementación del cliente

La aplicación cliente, al igual que el servidor, fue implementada en lenguaje *Java*. Como se dijo anteriormente, dado que la interfaz de invocación de servicios que provee el servidor es extremadamente simple, abierta y basada en tecnologías no propietarias, es posible implementar el servidor en otros lenguajes (incluso es posible implementar un cliente totalmente *web*). La decisión de utilizar *Java* como lenguaje para implementar el cliente ya fue justificada en la sección 4.4 (Diseño) de este documento.

El desarrollo del cliente presenta como mayor reto la implementación de la interfaz gráfica. En particular, la parte más compleja es la creación de un componente capaz de desplegar los resultados de una consulta en un árbol jerárquico. Dicho árbol debe ser navegable y capaz de acomodar conjuntos de resultados potencialmente enormes (del orden de miles). Para lograr este objetivo, se investigaron varias bibliotecas de visualización de grafos para el lenguaje *Java*.

Finalmente, se decidió basar la implementación del componente en cuestión en la biblioteca *Prefuse* (referirse a la sección Tecnologías utilizadas para más información). Algunas de las razones que influyeron en esta decisión fueron:

- *Prefuse* presenta una flexibilidad increíble a la hora de representar gráficamente una jerarquía, al permitir personalizar casi todos los posibles parámetros (organización de nodos, colores, tamaño, tipo de fuentes, etc.).
- Incluye un sistema completo de manejo de eventos en la representación gráfica, lo que facilita enormemente la implementación de un sistema de navegación.
- La biblioteca está diseñada para soportar jerarquías con miles de nodos sin necesidad de realizar trabajo adicional. La *performance* brindada por la librería es más que aceptable, incluso al visualizar jerarquías de gran tamaño.
- Presenta una *API* clara y extremadamente potente, lo que permite realizar extensiones y modificaciones libremente.
- Es de código libre y abierto (*OpenSource*).

Objetivos de la implementación del cliente:

El cliente implementado tiene como objetivo principal la demostración de la factibilidad de la metodología diseñada. La aplicación obtenida cumple a criterio de los desarrolladores con este objetivo.

Por otro lado, no se encuentra entre los objetivos de implementación del cliente la obtención de una aplicación adaptable a todas las necesidades. Por lo tanto, no es posible (al menos no sin un gran esfuerzo) adaptar la aplicación implementada más allá de cambios menores en la presentación y funcionalidades. La obtención de un cliente con características y funcionalidades realmente distintas requiere la implementación desde cero (o casi) de un nuevo cliente. Sin embargo, dicha tarea no debería resultar difícil ya que el servidor de búsqueda fue implementado teniendo en cuenta como uno de sus objetivos principales ser fácilmente "consumible" o utilizable. Dicho de otro modo, no hay nada de especial en la implementación del cliente creado, por lo que no es necesario basarse en este para la creación de uno nuevo.

Interconexión cliente – servidor:

La interconexión entre el cliente y el servidor está implementada en una clase (*DataAccess*) que funciona como un *Proxy*. Esto significa que dicha clase abstrae totalmente la interacción remota, haciendo aparecer al servidor como un objeto local para el resto de la aplicación cliente. Esto simplifica enormemente el desarrollo del cliente, ya que permite invocar desde cualquier punto de la aplicación a todos los servicios provistos por el servidor de búsqueda. La clase *DataAccess* fue implementada utilizando la librería *AXIS* (referirse a la sección tecnologías utilizadas para más información) para consumir el *WebService* publicado por el servidor de búsqueda.

Figura 4.50 Diagrama de clase DataAccess

Internacionalización del cliente:

El idioma de la interfaz gráfica del cliente puede ser cambiado en tiempo de ejecución gracias a la utilización de la tecnología de internacionalización provista por el lenguaje *Java*. Más aún, es posible añadir nuevos idiomas sin necesidad de modificar el código fuente de la aplicación.

Para añadir un nuevo idioma a los ya soportados por el cliente (español e inglés), es necesario únicamente traducir dos archivos de texto que contienen todos los strings utilizados por la interfaz gráfica de la aplicación. Estos archivos se encuentran en los paquetes *view* y *view.preferences*. Su nombre es *Strings.properties*.

Suponiendo que se desea añadir el idioma francés a los soportados por la aplicación, es necesario entonces traducir los archivos mencionados y guardarlos en los respectivos paquetes con el nombre *Strings_fr.properties*.

Se recomienda estudiar la documentación de la tecnología de internacionalización de J*ava* para más información sobre este punto.

Configuración del cliente:

Al igual que el servidor, el cliente utiliza un archivo de configuración. Los valores guardados en dicho archivo son modificados cuando el usuario realiza cambios en la configuración del cliente utilizando el panel de preferencias (ver apéndice E).

4.10 Definición de la base de datos

La base de datos de categorías fue diseñada con el objetivo de minimizar los tiempos de respuesta. Se decidió utilizar el manejador de bases de datos (*DBMS*) *MySQL* por ser un producto gratuito y con gran soporte en la comunidad *OpenSource*.

4.10.1 Importación de los datos

Se desarrolló una herramienta capaz de importar un conjunto de categorías a la base de datos automáticamente, lo que permite cargar el servidor con conocimiento sobre prácticamente cualquier dominio. El formato en el cual deben estar las categorías para ser importadas es extremadamente simple.

Proceso de importación:

Se toma como entrada un archivo conteniendo las categorías a importar. Dicho archivo debe contener un árbol que organice las categorías jerárquicamente. El siguiente ejemplo permite ver el formato:

```
Art
Art/Movies
Art/Movies/Titles
Art/Movies/Titles/The_Hours
Art/Movies/Titles/Star_Wars
```

Figura 4.51 Formato del archivo de la ontología

Cada línea del archivo contiene un camino desde la raíz hasta una hoja del árbol. Los nodos de los caminos deben estar separados por el carácter "/", y los espacios deben estar sustituidos por el carácter "_". En este contexto, se define una categoría como un camino desde la raíz a un nodo cualquiera del árbol. Por ejemplo, *Art* es una categoría y *Art/Movies/Titles* también. Una categoría es subcategoría de otra siempre que la segunda sea una subsecuencia de la primera (*substring*). Por ejemplo, *Art/Movies* es una subcategoría de *Art*, pero *Business* no lo es. Por más información sobre el proceso de importación ver el apéndice D.

4.10.2 Definición

Este es un punto crucial en lo que refiere al desempeño y eficiencia del sistema. Por tanto, fue un proceso evolutivo de definición y experimentación, hasta alcanzar al resultado definitivo.

Los problemas a resolver son los siguientes:

I. Dado un término, obtener todas las categorías en las que aparece el o una palabra con su misma raíz.
II. Dada una categoría *ODP*, obtener todas las categorías que surgen como sus sucesoras.

Inicialmente, estos problemas se resolvieron con dos tablas:

Tabla: **Topics**		
topicId : Valor Natural (int)	**path : Arreglo de caracteres (Varchar)**	topicTitle : Arreglo de Caracteres (Varchar)
Clave	Clave	

Los campos representan:

- **topicId:** identificador único del camino dado por una sucesión jerárquica de categorías *ODP*.
- **path:** sucesión jerárquica de categorías *ODP*.
- **topicTitle:** categoría de más especificidad en la jerarquía dada por **path**, procesada de la siguiente manera:
  - Se buscaban las raíces de las palabras que definían la categoría.
  - Se generaba una cadena de caracteres que tuviera todas las raíces obtenidas separadas por "_".

Tabla: **subTopics**		
path : Arreglo de Caracteres (Varchar)	**subTopicId : Valor Natural (int)**	topicId : Valor Natural (int)
Clave	Clave	Con Índice

Cuyos campos representan:

- **path:** sucesión jerárquica de categorías *ODP*.
- **topicId:** identificador del camino dado por la sucesión de categorías *ODP* que alcanza hasta el nivel anterior (padre) del camino dado en **path**.

- **subTopicId:** identificador único del camino determinado por las categorías *ODP* dadas en **path**. Estos son los mismos identificadores que aparecen en la tabla **Topics** con nombre **topicId**.

Con estas tablas, los problemas se resuelven de la siguiente manera:

I. Se buscaban todas las apariciones de la raíz (entre caracteres "_") del término ingresado. Se implementó de esta manera para evitar concordancias del estilo "*apple"* con raíz *appl* con el término *application* con raíz *applicat*.

Esta solución tiene algunos puntos en contra que resultan de importancia. Buscar la aparición de una cadena de caracteres dentro de otra, implica recorrer toda la cadena externa. Si a esto se le agrega que hay que buscarla en el campo **topicTitle** de todas las filas de la tabla **Topics**, se obtiene un cuello de botella en el desempeño del sistema.

II. A partir de una sucesión de categorías (**path**) se obtiene su identificador único en la tabla **Topics**, este es **topicId**. Luego se consiguen todos los caminos que tienen como "padre" a las categorías especificadas por **path.**

El campo **path** es clave de la tabla **Topics**, y los índices son implementados con árboles B. Por lo tanto, el costo (hablando sobre tiempo de ejecución) de obtención de **topicId** es bajo. Después, se encuentran todas las filas que contienen el identificador obtenido como **topicId** de la tabla **subTopics**. Como **topicId** tiene un índice en esa tabla, también se pueden obtener los ***paths*** en un tiempo relativamente pequeño.

Se considera que el problema *II* fue satisfactoriamente resuelto, pero sin embargo, no fue así con el *I*, por lo que, para resolver el problema *I* de una manera más conveniente, se agrega la siguiente tabla:

Tabla: **words**	
word : Arreglo de Caracteres (Varchar)	topicIds : Texto (text)
Clave	

Donde los campos representan:

- **word:** raíz de cada palabra que aparece en una categoría definida en *ODP*.
- **topicIds:** secuencia de identificadores de caminos *ODP* donde se puede encontrar un término con raíz **word**.

Entonces, el problema *I* pasa a resolverse con la siguiente estrategia:

I. Se busca en la tabla **words** la aparición de la raíz del término ingresado y se obtienen todos los identificadores de los caminos donde esa raíz está presente. Seguidamente, se adquieren los caminos en la tabla **topics** a partir de los identificadores conseguidos.

Esta nueva tabla evita tener que buscar en todas las filas de **topics** para encontrar las columnas deseadas. Sin embargo, como para algunas palabras la cantidad de caminos supera los 2000, el desempeño del sistema seguía siendo insatisfactorio.

Por tanto, se modificó tabla **words** de la siguiente forma:

Tabla: **words**	
word : Arreglo de Caracteres (Varchar)	paths : Texto (text)
Clave	

Donde el campo modificado tiene el sentido detallado a continuación:

- **path:** Secuencia de caminos *ODP* donde se puede encontrar un término con raíz **word**

Entonces el problema *I* se resuelve de la siguiente forma:

I. Se busca en la tabla **words** la aparición de la raíz del término ingresado y se obtienen todos los caminos donde esa raíz está presente.

De esta manera, con un único acceso a la base de datos, se obtienen todos los caminos donde aparece la raíz del término buscado.

Como se puede observar, la definición de las tablas fue una tarea de múltiples vaivenes y cambios. Sin embargo, se puede decir que actualmente el sistema guarda la información de manera que pueda ser obtenida en forma eficiente.

4.11 Ambiente de implantación y utilización masiva

La expresión "utilización masiva" se refiere a la implantación del sistema desarrollado en un ambiente donde la cantidad de usuarios sea muy elevada (del orden de decenas de miles de usuarios o más).

Sea cual sea la aplicación cliente/servidor que se desarrolle, la utilización en un ambiente de estas características requiere una infraestructura que en si misma es compleja, independientemente del servicio que se intente brindar.

Debido a que la aplicación desarrollada tiene como público objetivo personas sin grandes conocimientos sobre informática, es importante pensar en como se podría implantar un sistema de este tipo en un ambiente de "utilización masiva". Sin embargo, el alcance de este proyecto no es suficiente como para contemplar un problema de escalabilidad de este tipo.

De todas formas, el diseño tiene en cuenta (aunque no resuelve completamente) la escalabilidad por lo que a continuación se describe brevemente de qué forma es posible escalar el sistema y qué es necesario hacer.

4.11.1 Implantación en una red de área local (LAN)

En este caso, la implantación consiste en instalar el servidor en un *PC* (preferentemente dedicado si la cantidad de usuarios es elevada y se pretenden buenos tiempos de respuesta) que sea visible desde toda la *LAN* (o al menos desde las estaciones de trabajo que deseen utilizar el servicio). Posteriormente se debe instalar la aplicación cliente en todas las computadoras que deseen hacer uso del servicio y se debe configurar el acceso al servidor en las mismas. Dado que el tiempo de respuesta del servidor se encuentra en el orden de los segundos, soporta concurrencia y el ancho de banda de una *LAN* es normalmente suficiente, no es necesario hacer nada más.

El sistema no requiere mantenimiento de ningún tipo (salvo que se desee actualizar el conjunto de categorías, para lo que se provee de una aplicación de importación)

4.11.2 Implantación en la Web

En este punto es conveniente analizar dos casos diferentes de implantación del sistema:

- Cantidad de usuarios reducida. Podría ser el caso de que el software se venda, por lo cual quedaría disponible a un número limitado de usuarios.
- Uso irrestricto. Sería el caso de un software de libre distribución, donde estaría accesible a posiblemente millones de usuarios.

Cantidad de usuarios reducida:

Si la cantidad de usuarios es reducida, el único factor a tener en cuenta es el ancho de banda de subida con el que cuente el servidor. Si éste es suficiente, los tiempos de respuesta se verán limitados únicamente por el ancho de banda de bajada de los clientes.

El comportamiento del sistema sería similar al de una red de área local (*LAN*) donde no deberían presentarse problemas.

Uso irrestricto:

Si se quiere implantar la aplicación en la *Web* y que el servicio sea utilizado por cualquier persona, un único *PC* no podrá soportar la carga generada. En este caso la limitante no se encontrará únicamente en el ancho de banda, sino que el procesamiento se convertirá también en un cuello de botella. Para resolver esta situación, es necesario instalar varios servidores y utilizar un distribuidor de carga. Dicho distribuidor de carga no fue implementado, pero, dado que la interacción entre los clientes y servidores se lleva a cabo mediante el uso de *WebServices,* existen soluciones de fácil implementación para este tipo de problemas.

4.12 Verificación

Dado que este proyecto tiene como objetivo central estudiar la factibilidad de la metodología diseñada como herramienta de recuperación de información, la verificación de la implementación no fue una actividad prioritaria.
Los aspectos verificados fueron los siguientes: funcionamiento de la interfaz gráfica, testeo de caja negra de componentes, validación de correctitud de los resultados obtenidos y pruebas de carga del servidor.

4.12.1 Pruebas de carga del servidor

Es de vital importancia determinar la cantidad de consultas concurrentes soportadas por el servidor. Esto da la pauta de la cantidad de usuarios que puede soportar el sistema desarrollado. Para realizar estas pruebas se creó un simulador de clientes que emite consultas de forma concurrente a un ritmo configurable.

Plataforma en que corrió el servidor de búsqueda:

CPU	Intel Pentium 4 2.8 GHz
Memoria RAM	1 GB
Sistema operativo	Windows 2003 Server
JVM	Versión 1.4.2
DBMS	MySQL Server 4.1
Servidor de aplicaciones	Apache Tomcat 5.5.9

Figura 4.52 Computadora utilizada

Prueba de carga para soporte de pedidos concurrentes de categorías:

Estas pruebas tienen como objetivo medir el soporte a pedidos concurrentes de categorías del servidor de búsqueda. Los pedidos son únicamente del tipo consulta de categorías, en donde se le solicita al servidor que retorne un conjunto de categorías relacionado con una serie de términos. En esta prueba no se realizan pedidos de sugerencias ni de generación de consultas. El mecanismo utilizado es el siguiente:

1. Se crea un conjunto de procesos livianos (hilos o *threads*).
2. Cada uno de los hilos espera un intervalo aleatorio de entre 0 y 500 ms (milisegundos).
3. Cada uno de los hilos emite un pedido formado por un conjunto de entre 1 y 4 términos elegidos de manera aleatoria.
4. Se esperan dos segundos y se vuelve al primer punto.

La prueba se realizó de dos formas. Primero, los hilos de prueba fueron creados en una computadora conectada al servidor a través de *Internet* (128 Kbits de ancho de banda). Luego, fueron creados en otra computadora ubicada en la misma red (*LAN* de 100 Mbits de ancho de banda) que el servidor. La cantidad de hilos creados es un valor configurable, a mayor cantidad de los mismos, mayor carga de concurrencia. Los resultados obtenidos se pueden observar en la siguiente tabla:

Nº de hilos	Resultado obtenido (Local)	Resultado obtenido (*LAN*)
2	Todos los pedidos son atendidos en tiempo y forma.	Todos los pedidos son atendidos en tiempo y forma.
3	Algunos pedidos no son atendidos. Se satura el ancho de banda.	Todos los pedidos son atendidos en tiempo y forma.
30	No probado	Todos los pedidos son atendidos en tiempo y forma.
50	No probado	Al poco tiempo de comenzar la prueba, los pedidos dejan de ser atendidos. El servidor no resiste la carga (se queda sin memoria)

Figura 4.53 Resultados para la prueba de pedidos concurrentes de categorías

Prueba de carga para soporte de pedidos concurrentes de sugerencias:

En este caso, los pedidos son únicamente del tipo consultas de sugerencias, en donde se le solicita al servidor que retorne un conjunto de sugerencias en función de una serie de términos y un conjunto de categorías. En esta prueba no se realizan pedidos de categorías ni de generación de consultas. El mecanismo utilizado es el siguiente:

- Se crea un conjunto de procesos livianos (hilos o *threads*).
- Cada uno de los hilos espera un intervalo aleatorio de entre 0 y 500 ms.
- Cada uno de los hilos emite un pedido de sugerencias formado por entre 1 y 4 términos elegidos de forma aleatoria, más un conjunto de entre 1 y 4 categorías también seleccionadas de forma aleatoria.
- Se esperan dos segundos y se vuelve al primer punto.

La prueba se realizó de dos maneras. Primero, los hilos de prueba fueron creados en una computadora conectada al servidor por *Internet* (128 Kbits de ancho de banda). Luego, fueron creados en otra computadora ubicada en la misma red (*LAN* de 100 Mbits de ancho de banda) que el servidor. La cantidad de hilos creados es un valor configurable, a mayor cantidad de los mismos, mayor carga de concurrencia. Los resultados obtenidos se muestran a continuación:

Nº de hilos	Resultado obtenido (Local)	Resultado obtenido (LAN)
10	Todos los pedidos son atendidos en tiempo y forma.	Todos los pedidos son atendidos en tiempo y forma.
15	Algunos pedidos no son atendidos. Se satura el ancho de banda.	Todos los pedidos son atendidos en tiempo y forma.
20	No probado	Todos los pedidos son atendidos en tiempo y forma.
30	No probado	Todos los pedidos son atendidos en tiempo y forma.
40	No probado	Todos los pedidos son atendidos en tiempo y forma.
60	No probado	Todos los pedidos son atendidos en tiempo y forma.
80	No probado	Todos los pedidos son atendidos en tiempo y forma.
120	No probado	Todos los pedidos son atendidos en tiempo y forma.
200	No probado	El 10% de los pedidos no son atendidos. El resto de los pedidos son atendidos

		correctamente, pero en un tiempo considerablemente mayor. Aparentemente se satura la capacidad de procesamiento del servidor. La prueba debe ser abortada.

Figura 4.54 Resultados para la prueba de pedidos concurrentes de sugerencias

Prueba de carga para soporte de pedidos concurrentes de generación de consultas:

En este caso, los pedidos son únicamente del tipo generación de consultas para un motor de búsqueda, en donde se le solicita al servidor que retorne una consulta utilizable con un motor de búsqueda, a partir de un conjunto de términos, un conjunto de categorías y un conjunto de sugerencias. En esta prueba no se realizan pedidos de categorías o de sugerencias. El mecanismo utilizado es el siguiente:

- Se crea un conjunto de procesos livianos (hilos o *threads*).
- Cada uno de los hilos espera un intervalo aleatorio de entre 0 y 500 ms.
- Cada uno de los hilos emite un pedido eligiendo de forma aleatoria una serie de entre 1 y 4 términos, un conjunto de entre 1 y 4 categorías y otro conjunto de entre 1 y 4 sugerencias. Se esperan dos segundos y se vuelve al primer punto.

La prueba se realizó de dos formas. Primero, los hilos de prueba fueron creados en una computadora conectada al servidor por *Internet* (128 Kbits de ancho de banda). Luego, fueron creados en otra computadora ubicada en la misma red (*LAN* de 100 Mbits de ancho de banda) que el servidor. La cantidad de hilos creados es un valor configurable, a mayor cantidad de los mismos, mayor carga de concurrencia. A continuación se pueden observar los resultados obtenidos:

Nº de procesos	Resultado obtenido (Local)	Resultado obtenido (LAN)
5	Todos los pedidos son atendidos en tiempo y forma.	Todos los pedidos son atendidos en tiempo y forma.
10	Algunos pedidos no son atendidos. Se satura el ancho de banda.	Todos los pedidos son atendidos en tiempo y forma.
20	No probado	Todos los pedidos son atendidos en tiempo y forma.
30	No probado	Todos los pedidos son atendidos en tiempo y forma.
50	No probado	El servidor no soporta la carga. La prueba debe ser abortada.

Figura 4.55 Resultados para la prueba de pedidos concurrentes de generación de consultas

Test de carga para soporte de pedidos concurrentes de consultas mixtas:

En este caso, se realiza una combinación aleatoria de pedidos de los tres tipos anteriormente descritos. El mecanismo utilizado es el análogo al utilizado para cada tipo de pedido. Los resultados se muestran en la siguiente tabla:

Nº de procesos	Resultado obtenido (Local)	Resultado obtenido (LAN)
3	Todos los pedidos son atendidos en tiempo y forma.	Todos los pedidos son atendidos en tiempo y forma.
5	Algunos pedidos no son atendidos. Se satura el ancho de banda.	Todos los pedidos son atendidos en tiempo y forma.
20	No probado	Todos los pedidos son atendidos en tiempo y forma.
30	No probado	Todos los pedidos son atendidos en tiempo y forma.

50	No probado	Todos los pedidos son atendidos en tiempo y forma.
70	No probado	La carga no es soportada por el servidor. La prueba debe ser abortada.

Figura 4.56 Resultados para la prueba de pedidos concurrentes de consultas mixtas

Conclusiones de las pruebas de carga del servidor:

Las pruebas de carga del servidor muestra un buen soporte de concurrencia. Por ejemplo, el hecho de soportar 30 hilos en las pruebas realizadas significa que pueden ser atendidos 15 pedidos por segundo. Esto se traduce a aproximadamente 54.000 pedidos por hora o 1.296.000 pedidos diarios. Por otro lado, suponiendo que las consultas se distribuyen uniformemente a lo largo de l día y que un usuario realiza 50 pedidos por día, el soportar 30 hilos implica que sería posible atender 25000 usuarios.

Se puede ver que las pruebas soportan las afirmaciones realizadas sobre el desempeño del servidor en un entorno de uso masivo (ver sección 4.11: Ambiente de implantación y utilización masiva), al utilizar una conexión de bajo ancho de banda disminuye enormemente la capacidad de servicio del sistema. Dado que las consultas de categorías son las que más ancho de banda consumen, este tipo de pedidos son los más afectados.

Es importante notar que todas las pruebas fueron realizadas sin ajustar los parámetros de configuración del servidor de aplicaciones o las del manejador de base de datos más allá de lo normal. Es posible que manejando dichos parámetros se puedan obtener resultados aún mejores.

La prueba más importante es sin duda la prueba mixta, ya que realiza una combinación de operaciones similar a la realizada por los usuarios reales de la aplicación. Al realizar esta prueba con conexión de 100 Mbits, se puede ver que el servidor es capaz de atender sin problemas a un ritmo de 25 pedidos por segundo.

4.13 Experimentación

La etapa de experimentación es crucial en este trabajo ya que muestra la utilidad del sistema y las metodologías empleadas.

Resulta de importancia presentar los objetivos que se desean verificar en esta etapa del trabajo. Estos son:

- Obtener un sistema de generación de consultas. Esto implica:
  - Obtener el significado que el usuario desea.
  - Agregar términos que hagan más precisa la consulta.
  - Agregar sinónimos que ayuden a cubrir el área en forma más amplia.
- Alcanzar una interfaz simple y amigable.
- Obtener tiempos de respuesta cortos.

Estos objetivos y otros que no aparecen listados aquí ya fueron detallados en la sección 4.2.

El método de experimentación consistió en:

- Ofrecer en lenguaje natural la necesidad de obtener determinada información.
- Solicitar a diferentes usuarios que generen la consulta relacionada con esa necesidad.
- Enviar la consulta al buscador "*Google*".
- Observar cuántos resultados relacionados se encontraron entre los primeros 30.
- Contabilizar cuántos resultados fueron retornados en total.
- Invitarles a que generen ahora la consulta utilizando la aplicación desarrollada.
- Enviar la consulta generada por el programa a "*Google*".

- Verificar cuántos resultados relacionados se encontraron entre los primeros 30.
- Registrar cuántos resultados fueron retornados en total.
- Comparar resultados.
- Obtener tiempo de procesamiento del servidor.

De esta manera se puede observar fácilmente si los resultados obtenidos mejoran los ofrecidos inicialmente por el buscador. También es simple determinar si los tiempos de demora del servidor son aceptables o no.

Se hizo completar a los usuarios un formulario para obtener estos datos. La información concreta sobre los resultados obtenidos está en el apéndice F (Pruebas de usuarios).

Los usuarios fueron clasificados en tres clases:

- Experto: aquel que tiene experiencia en la utilización de buscadores y conoce y utiliza sus operadores avanzados. En general, sus consultas son bastantes largas (de más de 3 palabras).
- Nivel medio: es el que tiene algo de experiencia y casi no utiliza o directamente no conoce los operadores avanzados de los buscadores. Los largos de sus consultas suelen variar entre 2 y 4 palabras.
- Inexperto: tiene poca o directamente no tiene experiencia en la interacción con buscadores. Sus consultas contienen entre 1 y 3 palabras. También se engloba dentro de esta categoría a aquellos usuarios que expresan directamente la consulta en lenguaje natural al buscador y no utilizan únicamente las palabras más importantes.

A continuación se muestra una tabla con los valores obtenidos:

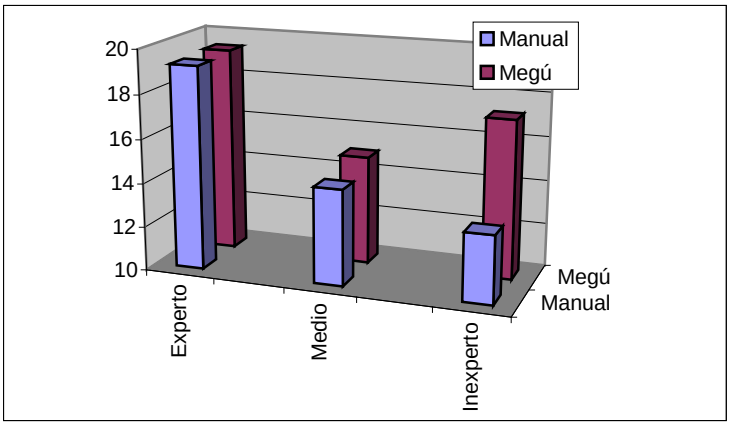

Figura 4.57 Resultados obtenidos

Se considera que los resultados no permiten inferir conclusiones sobre la utilidad del sistema. Para determinar su real utilidad deberían hacerse pruebas más extensas, esto se detalla como propuesta de trabajo futuro en la sección 6.1.5 (Evaluación del sistema desarrollado).

CAPÍTULO 5

5 CONCLUSIONES

La evaluación y obtención de resultados sobre el trabajo realizado debe realizarse a partir de los objetivos (sección 4.2) y la experimentación realizada (sección 4.13).

A continuación se analiza cada uno de los objetivos.

Sistema de generación de consultas:

Los experimentos realizados no permiten concluir resultados contundentes sobre la utilidad del sistema desarrollado. En parte porque los usuarios no se comprometieron lo suficiente con la actividad (esta resulta extremadamente tediosa). Además, la cantidad de relevamientos realizada es demasiado escasa como para realizar conclusiones estadísticamente significativas.

Teniendo en cuenta las condiciones comentadas, se extrae del relevamiento realizado que la aplicación parece resultar útil para usuarios inexpertos ya que mejora la cantidad de documentos relevantes obtenida.

En cuanto a los usuarios considerados "expertos" los resultados no son tan buenos. Los resultados no solo que no mejoran en valores relevantes, sino que además se consume más tiempo para realizar la consulta.

Más allá de las diferencias especificadas entre las capacidades de los usuarios, se puede establecer un umbral para las consultas generadas. Esto resulta más concreto en el momento de analizar la utilidad del sistema. Se dice que una consulta está generada correctamente cuando se obtienen al menos un 60 % de resultados relevantes dentro de los primeros 30 retornados.

Para consultas correctamente generadas no se obtienen mejoras considerables (incluso en algunos casos los resultados resultan menos precisos). En cambio, cuando las consultas no son generadas correctamente, los resultados resultan algo más alentadores. Es posible además, que afinando los parámetros estos resultados puedan ser mejores.

Ahora se pasa a comentar los ítems que componen este objetivo. Estos son:

- Obtener el significado que el usuario desea.
- Agregar términos que hagan más precisa la consulta.
- Agregar sinónimos que ayuden a cubrir el área en forma más amplia.

En general, sucede que cuando se logra obtener el verdadero sentido de la necesidad de información del usuario, se obtienen términos para expandir la consulta. Se dice que se obtiene el verdadero sentido de una consulta cuando la desambiguación de términos realizada resulta en los significados que el usuario estaba pensando cuando generó la consulta.

La tarea de obtener el verdadero sentido de la consulta es realizada a partir de:

- Los términos iniciales que el usuario ingresa.
- Las categorías que son retornadas de la ontología.
- La posterior selección de categorías que haga el usuario.

En algunas situaciones no se obtienen los significados deseados. Se concluye que se debe a alguno de los siguientes puntos:

- Los términos que utilizó inicialmente el usuario no se corresponden con los que se utilizan generalmente para describir los conceptos deseados.
- Las categorías realmente relacionadas con los términos especificados no se encuentran en la ontología cargada.
- Existen las categorías relacionadas pero también aparece una gran cantidad de resultados que dificulta encontrar estas.
- El usuario confunde conceptos y elige categorías no relacionadas.
- El diccionario que se utilizó no contempla el sentido deseado para el término ingresado.

En estos casos la aplicación puede llegar a desviar la consulta de la necesidad de información que se intentó especificar y hacer que los resultados sean peores.

Por lo tanto, es posible inferir que cuando se encuentra el sentido deseado para la consulta, o sea que la ontología contenía términos relacionados, y el diccionario contemplaba los verdaderos sentidos de los términos ingresados, entonces se ofrecen términos realmente relacionados y la consulta generada finalmente permite retornar mejores resultados.

Interfaz simple y amigable:

Dadas las funcionalidades provistas para la aplicación (explicadas en la sección 4.7.4: Diseño del cliente) y los comentarios recibidos por los usuarios de la interfaz en la fase de experimentación (sección 4.13), es válido afirmar que la aplicación resulta intuitiva de manejar. Al mismo tiempo, ayuda en su función principal que es proveer a un usuario inexperto de una herramienta para generar una consulta correctamente y encontrar la información deseada.

Introduciéndose en los ítems que componían este objetivo, se observa que:

- Los botones tienen dibujos explicativos de las funciones que cumplen y estos resultan intuitivos en el momento de ser utilizados.
- Las etiquetas informan de las funciones de los botones.
- Si bien la cantidad de opciones de visualización y navegación del árbol de resultados es bastante amplía (ver apéndice B: Interfaz gráfica) estas pueden comenzar a utilizarse a medida que se van conociendo ya que no son esenciales para el objetivo de la aplicación. En cambio, para el proceso de concebir la consulta, las funcionalidades provistas son bien concretas y no permiten desviarse de este objetivo.
- Las ayudas desarrolladas son de gran utilidad para la comprensión de la aplicación. La ayuda en línea explica la serie de pasos a dar para generar una consulta. La ayuda extendida permite ver cada funcionalidad al detalle.
- Se proveen dos idiomas para la interfaz de la aplicación (Inglés y Español) y un sistema simple para agregar nuevos idiomas a la aplicación.

Se considera que este objetivo fue completado satisfactoriamente.

Utilizar una ontología que categorice la información:

Se introdujo una ontología al sistema de manera que aporte información de relaciones entre conceptos. Esta permite relacionar términos y se complementa muy bien con un diccionario en la tarea de clasificar y desambiguar términos.

División de tareas entre cliente y servidor:

Se considera que esta trabajo fue correctamente desempeñado ya que se cumplió con sus principales metas:

- Evitar la instalación por parte de un usuario común de un manejador de bases de datos y un diccionario como *WordNet*.
- Tener la capacidad de actualizar la ontología y el diccionario utilizados sin que haya que volver a instalar la aplicación.
- Modificar parámetros del sistema sin que se vean afectados los usuarios.

Al necesitar la base de datos y el diccionario únicamente del lado del servidor, el usuario de la aplicación cliente se evita todos esos problemas.

Existía otra disyuntiva relacionada con las tareas de visualización de los datos. Finalmente, como se explica en la sección 4.7.4: Diseño del cliente, el procesamiento necesario para implementar esta funcionalidad recae completamente del lado del cliente. Esto evita una mayor sobrecarga del servidor y utiliza mejor los recursos que dispone el cliente.

Algoritmos eficientes y con tiempos de respuesta cortos:

Se puede observar en las pruebas realizadas que los algoritmos desarrollados no presentan retardos en las condiciones establecidas.

Se trabajó fuertemente en la reducción de tiempos de ejecución para los algoritmos utilizados. Los esfuerzos fueron fructíferos como se puede ver en los resultados de las pruebas de carga del servidor.

En la sección 4.12 (Verificación) se realizaron pruebas de carga sobre el sistema y ahí se vislumbra que en condiciones de escaso ancho de banda se pueden presentar ciertos retrasos que pueden llegar a molestar al usuario que utiliza el programa. Cuando el ancho de banda es suficiente, los resultados son más que satisfactorios.

La propuesta es, como se verá en la sección 6 (Trabajos futuros), establecer un sistema distribuido con un manejador de carga que derive las consultas a distintas máquinas de forma de no recargar demasiado estas y obtener resultados satisfactorios.

Estudio de nuevas metodologías:

Realizando un análisis del sistema desarrollado, se puede ver que la metodología planteada es generalizable. Dado un conjunto de documentos (repositorio de información), una ontología que jerarquiza los tipos de documentos presentes en el repositorio y un diccionario, la metodología consiste de los siguientes pasos:

- El usuario formula una consulta consistente de términos
- Se busca en la jerarquía u ontología del dominio los posibles tipos de documentos relacionados con la necesidad de información del usuario.
- Se le presentan al usuario los posibles tipos de documentos relacionados ordenados jerárquicamente.
- El usuario marca los tipos de documentos que le resultan relevantes.
- Utilizando la información de los tipos de documentos relevantes, se intenta desambiguar los términos buscados.
- Utilizando el diccionario, se le presentan al usuario términos relacionados fuertemente con los que especificó inicialmente.
- El usuario selecciona términos relacionados.
- Se formula una nueva consulta en base a la información recabada. Esta consulta es utilizada por un motor de búsqueda estándar.

En la implementación desarrollada se instanció cada uno de los elementos descritos (repositorio, ontología, diccionario y motor de búsqueda) mostrando la factibilidad de el método planteado.

La metodología parece ser innovadora ya que no se encontró información sobre trabajos que describan sistemas de estas características.

Sistema flexible a cambio e inserción de nuevas ontologías:

El sistema desarrollado utiliza la ontología de manera genérica, ya que únicamente requiere que esta esté organizada jerárquicamente. Cualquier conjunto de categorías jerarquizado puede ser importado sin problemas a la base de datos utilizada por el servidor de búsqueda. Para comprender el proceso de importación ver el apéndice D (Importador de base de datos).

Sistema flexible a cambio del diccionario:

Como se dijo anteriormente (sección 4.9.1: Implementación del servidor), no se logró independencia del diccionario utilizado (*WordNet*). Esto se debió principalmente a que no existen otros diccionarios que ofrezcan funcionalidades equivalentes, como por ejemplo: obtención de sinónimos, hiperónimos e hipónimos. Estas características avanzadas de *WordNet* son las que permiten realizar los cálculos necesarios para implementar la metodología desarrollada.

Flexibilidad a cambios en el idioma de dominio de la aplicación:

Para cambiar el idioma del dominio de la aplicación es necesario cambiar la ontología y el diccionario. Aunque es factible encontrar buenas ontologías para varios idiomas, el único diccionario de uso libre suficientemente poderoso como para obtener resultados positivos, es la versión en inglés de *WordNet*. El cambio de idioma depende entonces de la existencia de un diccionario en el idioma adecuado que presente las mismas funcionalidades que la versión en inglés de *WordNet*.

CAPÍTULO 6

6 TRABAJO FUTURO

Si bien el proyecto se considera un trabajo terminado. Hay problemas que podrían haberse resuelto de manera más elegante. Hay soluciones que no son las mejores y que de haber contado con una mayor cantidad de tiempo, se hubieran visto modificadas.

A continuación se detallan los ítems que pueden ser mejorados y extendidos en el trabajo realizado.

6.1 Extensiones al servidor

El servidor ofrece algunas funcionalidades a ser mejoradas. Estas son explicadas en los siguientes puntos.

6.1.1 Selección de significados para los términos

Como se explica en la descripción del sistema (sección 4.4) y más detalladamente en el Apéndice A (Algoritmos), una vez elegidas las categorías deseadas, se trata de desambiguar los conceptos y ofrecer términos relacionados. Luego de eso, puede suceder que el usuario no esté de acuerdo con el significado elegido para un término.

Como está implementado el sistema hasta el momento, en esta situación el usuario tiene la opción de elegir nuevas categorías para que el sistema elija correctamente el significado, o de lo contrario, no elegir ninguna sugerencia para realizar la expansión de la consulta. Sin embargo, esto puede ser resuelto de manera más elegante y provechosa para el usuario (aunque con mayor sobrecarga del servidor), ofreciendo las restantes definiciones para el término desambiguado. De esta forma, se podría modificar el sentido elegido y obtener nuevas sugerencias.

Para poder implementar este servicio, habría que agregar una nueva funcionalidad al servidor. Esta tendría la función de retornar todas las definiciones encontradas para un término dado.

Esto implica modificaciones en el cliente, por lo tanto, este punto será tratado nuevamente en la Sección 6.2.1 (Extensiones al cliente).

6.1.2 Utilización de información semántica

Existe la posibilidad de modificar el algoritmo de selección de significados para los términos utilizando información semántica obtenida de la interacción con el usuario.

La idea no fue estudiada en el marco de este trabajo, pero se pueden utilizar herramientas de análisis semántico para obtener sugerencias a las palabras introducidas, e intentar mejorar los resultados obtenidos.

La propuesta es generar un conjunto de reglas semánticas (utilizando *YACC* o *CUP*) para determinar la forma gramatical en que se utilizaron las palabras ingresadas en la consulta inicial. De esta manera se reduce la cantidad de significados para elegir en cada término, aumentando la precisión obtenida.

6.1.3 Preprocesamiento de significados de los términos

Cuando se desean obtener sugerencias para un término dado, se realiza un procesamiento importante implicado por la necesidad de generar conjuntos de palabras que representen cada significado. Esta tarea se realiza cada vez que se requiere desambiguar un término. Sin embargo, se podría realizar un preprocesamiento para los sentidos de cada una de las palabras que contiene el diccionario (*WordNet*). Esto se puede realizar ya que no depende de ninguna información que puede verse modificada más allá de la palabra en cuestión.

Como se puede ver en el Apéndice A, el conjunto de palabras obtenido pondera cada una de ellas con un valor de relevancia para el significado dado. Además, agrega las palabras que forman parte del significado de cada uno de sus hiperónimos hasta cierto nivel especificado.

Guardando la información de cada sentido de los términos del diccionario, y guardando además el valor con el cual se ponderó cada palabra, se logra evitar este procesamiento en tiempo de ejecución.

Para esto se propone la generación de la siguiente tabla en la base de datos:

Tabla: **definitions**	
sense : Valor Natural (int)	words : Texto (text)
Clave	

Cuyos campos representan:

- **sense:** identificador del sentido dado por el diccionario a una palabra.
- **words:** secuencia de elementos de la forma:
  - raíz de la palabra
  - importancia otorgada

  Esta secuencia de elementos estaría separada por algún carácter especial.

A continuación se muestran algunos ejemplos sobre como quedarían representados los datos en las filas de la tabla:

Tabla: **definitions**	
sense : Valor Natural (int)	words : Texto (text)
topic	subject+20|theme+20| issue+20|mather+20|message+18|content+18| communication+16|
house	home+20|menage+20|domicile+20|abode+18|building+18|edificie+18| structure+16
cook	fix+20|ready+20|make+20|prepare+20|create+18|change+18| modify+16|

En el ejemplo se utilizaron las siguientes presunciones:

- El carácter "+" separa un término de su valor de relevancia.
- EL carácter "|" separa conjuntos término-relevancia.
- El máximo valor de relevancia utilizado es 20.
- Se decrementa el valor de relevancia de 2 en 2 a medida que se avanza por la relación de hiperonimia.

En caso de que el diccionario a utilizar sea diferente de *WordNet*, se debería agregar otra tabla que especifique los identificadores de los sentidos de cada palabra:

Tabla: **wordsenses**	
word : Arreglo de Caracteres (Varchar)	ids : Texto (text)
Clave	

Donde los campos representan:

- **word:** palabra a desambiguar.
- **ids:** secuencia de elementos que contiene los identificadores de los sentidos de la palabra **word**.

Con estas tablas, se podría obtener la información deseada de la siguiente forma:

- Utilizando la palabra de la cual se quieren obtener los sentidos, se busca en la tabla **wordsenses** los identificadores de estos sentidos.
- A partir de la cadena de caracteres (separada por algún carácter especial) se forma una colección de identificadores.
- Para cada identificador:
  - Se obtiene de la tabla **definitions** la cadena de caracteres con las palabras y sus valores.
  - Se arma una colección de palabras y valores.

Esta información es exactamente la que se obtiene procesando cada palabra en tiempo de ejecución.

6.1.4 Recolección de información estadística

Se puede ver en numerosos ámbitos de la informática (y no sólo de ella) que la recolección de información suele resultar provechosa para, ya sea mejorar el desempeño de un sistema, o de una empresa, o simplemente evaluar como este se comporta frente a los deseos del usuario.

Puede resultar interesante entonces, recolectar cierta información estadística. Por ejemplo:

1. Búsquedas ingresadas
2. Largo de estas búsquedas
3. Sugerencias elegidas
4. Buscadores utilizados
5. Utilización de la ayuda en línea
6. Utilización de la ayuda extendida
7. Perfiles de usuario que se generan

La información obtenida de cada uno de estos ítems tiene variadas aplicaciones. Se pueden destacar entre ellas:

1. Búsquedas ingresadas:

Resulta de interés guardar información sobre las búsquedas generadas ya que puede utilizarse para ver:

- Intereses de los usuarios: cuales son las áreas que le interesan (más allá de lo que pueda informar un perfil de usuario), así como la clase de consultas que realiza diariamente (noticias, deportes, etc).

- Estructura de las búsquedas: como conforma las búsquedas para obtener los resultados deseados. Si utiliza mayoría de sustantivos o no. Si modifica su comportamiento luego de utilizar el sistema, que puede ser señal de comenzar a comprender como funcionan (o mejor dicho, como se comportan) los buscadores.

2. Largo de las búsquedas:

Datos importantes pueden extraerse del largo de las consultas. Esto muestra, de alguna manera, la práctica que tiene el usuario en la búsqueda de información. Mientras más palabras agrega a la consulta, más conoce el comportamiento de los buscadores y más claro tiene la información que quiere encontrar. También resulta interesante observar si modifica su comportamiento mientras más frecuente es la utilización del sistema.

3. Sugerencias elegidas:

Esto expone la capacidad del sistema de encontrar el sentido deseado para un término, si el usuario elige términos relacionados, de alguna forma quiere decir que el sentido elegido es, al menos, similar al deseado por el usuario.

Además, es posible presentar un nuevo ordenamiento de los sentidos que se les da a los términos ingresados. Si el programa elige correctamente de manera sistemática el sentido de las palabras, entonces se puede inferir que cuando el usuario ingresa determinadas palabras (y elige o no algunas categorías) estaba pensando sobre ese sentido del término. En este caso, se podría modificar el sistema para que de mayor importancia a los significados anteriormente seleccionados, frente a la elección del sentido de una palabra frente a un nuevo usuario.

También se podría utilizar para reordenar las sugerencias de manera de hacer aparecer más arriba las que más veces son elegidas.

4. Buscadores utilizados:

El sistema (por defecto) ofrece la posibilidad de elegir tres buscadores:

- Google
- Yahoo
- Altavista

Se puede guardar información de cuáles son los más utilizados para determinar cual está otorgando los mejores resultados.

Hilando un poco más fino en este punto, resulta de interés observar cuáles buscadores son más utilizados en referencia con diferentes temáticas (dadas por la información inherente a la consulta ingresada y/o por el perfil del usuario). Y por tanto, tratar de ofrecer al usuario, un buscador diferente dependiendo de cual sea el área en que es posible englobar la consulta deseada.

5. Utilización de la ayuda en línea:

Como se dijo anteriormente, este es un sistema de ayuda al usuario inexperto. Por tanto, es interesante saber cuantas veces debe utilizar el programa con la ayuda que explica cada paso a realizar para generar una búsqueda. Esto sirve para ver que tan inexperto es el usuario, y que tan clara es la interfaz como para utilizarla sin una guía.

6. Utilización de la ayuda extendida:

Al igual que en el punto anterior, se puede verificar observando la cantidad de veces que esta es utilizada, la claridad con que fue diseñada la interfaz.

A su vez, se puede determinar la utilidad de la ayuda en línea.

7. Perfiles de usuario que se generan:

El perfil de usuario aporta información importante para guardar y luego utilizar de diferentes maneras.

Esta es una lista de algunos de los datos que ofrece:

- Informa de los intereses de los usuarios.
- Utilizado en conjunción con las consultas ingresadas, informa que tan relacionadas se encuentran estas con el perfil marcado. Además muestra cuantas veces se busca información relacionada con sus intereses y cuantas no.
- Utilizado con las categorías seleccionadas, se puede ver que tanto se eligen categorías que estén relacionadas con el perfil especificado.

Toda esta información puede ser manipulada en el futuro para dar mayor o menor importancia al perfil en la selección de categorías, sentidos de las palabras, términos relacionados, etc.

6.2 Extensiones al cliente

El cliente también ofrece algunas funcionalidades a ser mejoradas. Estas se exponen en los siguientes puntos.

6.2.1 Selección de significados para los términos

Como se comentó en la sección relativa a las extensiones del servidor (Sección 5.1), también debe ser modificado el cliente en caso de querer ofrecer la posibilidad de modificar el sentido elegido por el sistema para un término dado.

Debería agregarse un botón para obtener todos los sentidos de un término. Luego hay que implementar la llamada al servidor que retornara la información en un formato comprimido. Esta debería ser transformada en información comprensible para el usuario. Además tendría que ser mostrado de manera que el nuevo sentido pueda ser fácilmente elegido. A continuación, se debería pedir la lista de palabras relacionadas con ese nuevo significado que luego se mostraría en las ventanas ya existentes para ese propósito.

6.2.2 Migración del cliente a una interfaz web

El cliente fue desarrollado en el lenguaje *Java* como una aplicación de escritorio. Sería extremadamente interesante implementar el mismo como una aplicación enteramente *web*, lo cual evitaría la necesidad de instalar software en la máquina del usuario.

Para realizar esta transformación existen diferentes alternativas manejadas en la sección 4.8.2 (Cliente).

6.2.3 Búsqueda en múltiples servidores

Actualmente el cliente obtiene los resultados de un único servidor (el cual es configurable). Una posible extensión es que el cliente pueda buscar simultáneamente en varios servidores y que los resultados sean fusionados y visualizados a la vez.

Esto resultaría útil en caso de que existan diferentes servidores con distintas ontologías. De esta manera, se enviaría la misma consulta a estos servidores, se obtendrían los conjuntos de categorías de cada servidor, y luego se generaría un único árbol de resultados fusionando estos conjuntos.

6.3 Desambiguación de palabras

En el trabajo realizado, se utilizó una metodología para desambiguar palabras que puede obtener buenos resultados (ver sección 4.4: Descripción del Sistema). La idea es presentar la posibilidad de seguir trabajando con esta metodología dentro del área y observar otras aplicaciones que esta pueda tener.

6.3.1 Detección del significado de un término

Como ya se vio en la descripción del sistema (sección 4.4), el método para obtener el significado de un término es el siguiente:

- Se buscan categorías relacionadas en la ontología.
- Se seleccionan las que están relacionadas.
- Se desambigua la palabra a partir de las categorías elegidas y sus ancestros en la jerarquía.

Este método para sustituir la selección del significado de términos de un diccionario. En vez de elegir el significado de las opciones que retorna un diccionario (se recuerda que *WordNet* retorna 34 sentidos para la palabra *work*) se pueden elegir términos dentro de una ontología y dejar que el sistema la desambigüe.

Si bien, el sistema propuesto no es exacto, se considera que generando una ontología de propósito general, a partir de las definiciones de un diccionario, se pueden obtener resultados muy satisfactorios.

6.3.2 Traducción de textos

A partir del punto desarrollado en el paso anterior, se propone la metodología para generar un traductor de textos interactivo.

Este se detendría en cada palabra ambigua y presentaría términos relacionados jerárquicamente para desambiguarla. De esta manera se ayudaría en la traducción de un texto utilizando una ontología de propósito general.

También existe la posibilidad de generar un texto anotado con cada palabra ambigua de manera que el traductor utilice esta información para obtener el correcto sentido de las palabras y poder realizar su trabajo de manera correcta.

GLOSARIO

.NET: Es una plataforma (similar a *Java*) de desarrollo y ejecución de aplicaciones. Esta permite combinar fácilmente programas hechos en diferentes lenguajes de programación (como *C*, *C++*, *Visual Basic*) y obtener un único sistema que utilice los demás a su servicio. Está en continuo desarrollo y cada vez ofrece más funcionalidades.

AJAX: Son un conjunto de tecnologías que establecen un punto medio entre las aplicaciones de escritorio y los clientes enteramente *web* (que se ejecutan en el navegador). Ofrecen la posibilidad de migrar parte del procesamiento que se hace en el servidor para los clientes *web* utilizando el lenguaje de programación *Javascript* que se ejecuta en el navegador. También ofrecen la posibilidad de representar dinámicamente los datos mediante un modelo llamado *Document Object Model*. Este modelo permite el acceso y la actualización dinámica de contenido estructural o del estilo de la página. El sitio oficial del *Document Object Model* es: http://www.w3.org/DOM/

Algoritmo de Porter: Es la especificación de un sistema para obtener raíces de palabras (stemmer). El algoritmo de *Porter* está disponible aquí: http://www.tartarus.org/~martin/PorterStemmer/.

API: Una *API* (*Application Programming Interface*) o interfaz para programación de aplicaciones es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

Apple Macintosh: Empresa norteamericana dedicada a la producción de hardware para computadoras. Su sitio oficial en *Internet* es: http://www.apple.com/.

Applet: Es un programa escrito en *Java* que tiene la interesante característica de poder ser incluido en una página *web* de la misma manera que es incluida una imagen en estos sitios.

ASP: Tecnología desarrollada por *Microsoft* para sus servidores de páginas *web*. Una página *ASP* es una página *HTML* que incluye uno o más scripts.

AXIS: Es un motor para *SOAP*. Permite construir procesos que intercambien mensajes a través de *SOAP*, como clientes y servidores.

BeOS: Es un sistema operativo para PC de origen francés desarrollado por *Be Incorporated* en 1990, orientado principalmente a proveer alto rendimiento en aplicaciones multimedia.

Buscador: Es un sistema de *Recuperación de Información* orientados a un entorno *web*. Generalmente consta de una interfaz de consulta desplegada a través de un navegador.

CiteSeer: También conocido como *Research Index*, es un servicio de búsqueda de documentos de investigación. El sitio web es: http://citeseer.ist.psu.edu/cs.

Clustering: Métodos de agrupación de documentos relacionados por una misma temática o área de interés.

Consulta sincrónica: Es una clase de pedidos caracterizada por que el que efectúa el pedido queda esperando la respuesta y no puede realizar otra acción hasta que esta no llegue.

CORBA: *CORBA* (*Common Object Request Broker Architecture*) es la especificación de una arquitectura estándar para *ORB* (*Object Request Broker*). Esta es una tecnología que maneja la comunicación y el intercambio de datos entre dos objetos, sin importar si se encuentran en la misma máquina o conectados a través de una red. *ORB* promueve la interoperabilidad entre objetos en sistemas distribuidos.

CUP: Es una herramienta para generar analizadores semánticos escrita en *Java*. El código resultante es también código *Java*.

DBMS: Es la sigla que denomina a los manejadores de bases de datos relacionales (*Database Management System*).

Escalabilidad: Cualidad que se mantiene cuando el sistema continúa satisfaciendo sus requerimientos en situaciones en que sus parámetros se han incrementado.

Facade: Patrón de diseño que especifica como hacer que dos subsistemas interactúen. Este aconseja utilizar una única clase que actúe como fachada entre los dos subsistemas.

Firewall: Sistema diseñado para prevenir el acceso no autorizado a o desde una red privada. En general se utilizan para prohibir el acceso a una máquina de intrusos en Internet.

Hiperonimia: Es una relación de subordinación entre términos. Se dice que un término es hiperónimo de otro si el último es una clase del primero.

Hiponimia: Es la relación opuesta a la hiperonimia. Se dice que un término es hipónimo de otro si el primero es una clase del último.

Holonimia: Es la relación opuesta a la meronimia. Es el término que define el todo en la dependencia.

Homonimia: Es una cualidad que se le da a un término cuando tiene la facultad de representar diferentes conceptos.

HTML: El *Hyper Text Markup Language* es un lenguaje informático diseñado para estructurar textos. Este es el formato estándar de las páginas web y el que actualmente comprenden los navegadores.

HTTP: Es un protocolo para transmisión de datos. Es el más difundido desde la creación de *Internet*. El sitio oficial es: http://www.w3.org/Protocols/.

Internacionalización de Java: *Java* permite traducir una interfaz a diferentes idiomas definiendo un archivo de propiedades de la aplicación. Por más información se puede acceder al sitio *web*: http://java.sun.com/developer/technicalArticles/Threads/swing/?frontpage-jdc#int

Internet: Es una red de redes a escala mundial. Consiste de millones de computadoras interconectadas por un mismo protocolo de comunicación.

Interoperabilidad: Es la habilidad de dos o más computadoras para trabajar juntas y comunicarse.. Estas pueden tener diferentes arquitecturas y/o sistemas operativos.

IR: Es la sigla que identifica el concepto: *Recuperación de Información* (*Information Retrieval*).

J2SE: La *Java* 2 *Standard Edition* es una plataforma de desarrollo utilizada para proyectos de pequeño y mediano porte llamada también *JDK* (*Java Development Kit*). Además contiene la misma versión de la máquina virtual de *Java* llamada *JRE* (*Java Runtime Environment*).

Javascript: Es un lenguaje de programación derivado de *Java*. El código se ejecuta en el navegador a través de la *JRE*.

JRE: Esta es la máquina virtual de *Java* (*Java Runtime Environment*) encargada de interpretar el código escrito para ese lenguaje y traducirlo al código de la máquina subyacente. El sitio oficial para obtenerla es: http://java.sun.com/j2se/1.4.2/download.html.

JSP: Es la tecnología (*Java Server Pages*) para generar páginas *web* de forma dinámica en el servidor, desarrollado por *Sun Microsystems*. Está basado en *scripts* que utilizan el lenguaje *Java*.

LAN: Es una Red de Área Local (*Local Area Network*). Se refiere a redes locales de computadoras.

Lematizador: Es un sistema que obtiene una palabra en su forma más simple. Esta es la que aparece como entrada en diccionarios y tesauros.

MacOS: Es el sistema operativo de la compañía *Apple Computer* para el ordenador personal *Macintosh*, aparecido en 1984. Este sistema fue concebido originalmente como sistema de ventanas que ha servido de modelo para otras plataformas.

Meronimia: Es la relación semántica entre un término que define una parte y un término que define un todo. El merónimo es el término que define la parte en la dependencia.

Metabuscador: Servicio disponible para interactuar con más de un buscador utilizando una única interfaz de consulta.

MySQL: Es el manejador de bases de datos (*Database Management System – DBMS*) gratis y de código abierto (*OpenSource*) para *SQL* más popular del mundo.

ODP: El *Open Directory Project* es un proyecto que intenta categorizar las documentos disponibles en *Internet*. Establece una jerarquía de categorías donde se insertan manualmente páginas de *Internet*. Estas categorías están también definidas por humanos.

Su base de datos es actualizada semanalmente con nuevas categorías y páginas *web*.

Ontología: Es una construcción que estructura contenidos explícitos. Esta jerarquiza términos y define relaciones entre ellos.

OpenSource: Refiere a proyectos abiertos al público en general. Generalmente se utiliza para proyectos relacionados con la computación donde determina además que el código fuente del software desarrollado está disponible.

Página web / Sitio web: Es como se les llama vulgarmente a los documentos disponibles en *Internet*. Estos se encuentran alojados en alguna de las máquinas que forman parte de la red.

Perl: El *Practical Extraction and Report Language* es un lenguaje de programación desarrollado por *Larry Wall*. Esta inspirado en otras herramientas de *UNIX* como son: *sed*, *grep*, *awk*, *c-shell*, para la administración de tareas propias de sistemas *UNIX*. También ha sido ampliamente difundido para la generación de páginas *web* dinámicas.

PHP: Es un lenguaje de programación de *scripts*. Se utiliza principalmente para la programación de páginas *web* dinámicas, destaca por su capacidad de ser embebido en el código *HTML*.

Polisemia: Es la propiedad de un término de tener varios significados.

Precisión: Es la cantidad de documentos relevantes recuperados sobre el total de resultados obtenidos.

Prefuse: Es un conjunto de herramientas para crear visualizaciones altamente interactivas de conjuntos de datos estructurados y no estructurados.

Recall: Es la cantidad de documentos relevantes recuperados sobre el total de documentos relevantes.

Relevancia: Coherencia de los resultados recuperados respecto a las necesidades de adquisición de información del usuario.

Script: Es un programa o una secuencia de instrucciones interpretada por otro programa en lugar de ser procesada directamente por el procesador de la computadora.

Sinonimia: Es la relación que vincula términos con un mismo significado.

SOAP: El *Simple Object Access Protocol* es un protocolo simple de intercambio de información a través de Internet basado en *XML*.

SQL: El lenguaje de consulta estructurado (*Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Reúne características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés, en forma sencilla, de una base de datos.

Stemmer: Es una herramienta que obtiene la forma de una palabra luego de que todos sus afijos (prefijos y sufijos) fueron removidos.

String: Cadena de caracteres.

Tesauro: Es una clase de diccionario que contiene términos del lenguaje natural y su equivalencia con los términos normalizados del lenguaje documental.

Tomcat: Es un contenedor de aplicaciones Java de tipo servidor. Encapsula aplicaciones que van a interactuar con diferentes clases de clientes.

Web Service: Es una aplicación para un entorno *web* que utilizan estándares basados en *XML* y protocolos de transporte para intercambiar datos con diferentes aplicaciones cliente.

Web: También llamada *World Wide Web* (*WWW*) es el conjunto de computadoras en el que se almacena la información que se encuentra en *Internet*.

WordNet: Es un diccionario o sistema de referencia léxico. Los significados de los términos están representados en conjuntos de sinónimos (*Synsets*) que son listas de palabras intercambiables en algún contexto.

WSDL: Este es un lenguaje para describir servicios *web* como un conjunto de funciones o funcionalidades que provee un servidor determinado. Esta descripción se realiza en formato *XML*.

XML: El *Extensible Markup Language* es un lenguaje de descripción de páginas diseñado con la intención de reemplazar al estándar *HTML*. Este permite personalizar las etiquetas que

describen la presentación y el tipo de los elementos de datos.

YACC: Es una herramienta para generar analizadores semánticos disponible en los sistemas *UNIX*. Su nombre es acrónimo de *Yet Another Compiler Compiler*.

REFERENCIAS

[Pink94] Brian Pinkerton, *"Finding What People Want: Experiences with the WebCrawler"*, Second International WWW Conference, 1994.

[Sant03] Santamaría - Gonzalo - Verdejo, "*Automatic Association of Web Directories to Word Senses*", Association for Computational Linguistics, 2003

[Guer99] Guerrero Bote - Lozano Tello, "*Vínculos entre las Ontologías y la Biblioteconomía y Documentación*", Granada, ISKO-Facultad de Biblioteconomía y Documentación, (Pág: 25-31), 1999.

[Nech91] R. Neches, "*Enabling technology for knowledge sharing*", AI Magazine vol. 12, nº 3 (Pág: 36-56), 1991.

[Port80] M. F. Porter, "*An algorithm for suffix stripping*", Program 14 nº 3 (Pág 130-137), 1980, http://www.tartarus.org/~martin/PorterStemmer/

[QinP00] J. Qin - S. Paling, "*Converting a controlled vocabulary into an ontology: the case of GEM*", Information Research vol. 6, nº 2, 2000-01, http://informationr.net/ir/6-2/paper94.html

[Ding02] Y. Ding - S. Foo, "*Ontology research and development. Part 2-a review of ontology map-ping and evolving*", Journal of Information Science vol. 28, nº 5 (Pág: 375-388) 2002.

[Voor98] E. Voorhees, "*WordNet, an electronic Lexical Database*", Mit Press, 1998.

[Mand98] R. Mandala - T. Takenobu - T. Hozumi "*The use of Wordnet in information retrieval*", Proceedings of Coling-ACL, 1998.

[Baez99] Ricardo Baeza - Yates. *"Modern Information Retrieval",* Addison Wesley, 1999.

[Bhat92] Sanjiv K. Bhatia, "*Selection of Search Terms based on User Profile*", Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing: technological challenges of the 1990's (Pág: 224-233), 1992.

[Chen93] Hsinchum Chen – Linlin She, *"An Inductive Machine Learning Approach to Information Retrieval"*, Technical Report – Artificial Intelligence Lab. – University of Arizona, 1993, http://ai.bpa.arizona.edu/html/papers.html.

[Efth93] Efthimis N. Efthimiadis, *"A user-centred evaluation of ranking algorithms for interactive query expansion",* Annual ACM Conference on Research and Development in Information Retrieval (Pág: 146-159), 1993.

[Hoas00] Keiichiro Hoashi - Kazunori Matsumoto - Naomi Inoue - Kazuo Hashimoto, *"Document Filtering Method Using Non-Relevant Information Profile"*, Annual ACM Conference on Research and Development in Information Retrieval (Pág: 176-183), 2000.

[Joac02] Thorsten Joachims, *"Optimizing Search Engines using Clickthrough Data"*, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (Pág: 133-142), 2002.

[Korf84] Robert R. Korfhage, *"Query enhancement by user profiles"*, Conference on Research and Development in Information Retrieval (Pág: 111-121), 1984.

[Koyc01] Ivan Koychev, *"Learning about User in the Presence of Hidden Context"*, In Proceedings of Machine Learning for User Modeling, 2001.

[Kwon96] Oh Woog Kwoni - Myoung Cheol Kim - Key Sun Choi, *"Expansion Using Domain-Adapted, Weighted Thesaurus in an Extended Boolean Model"*, Conference on Information and Knowledge Management (Pág: 140-146), 1996.

[Mand99] Rila Mandala – Takenobu Tokunaga – Hozumi Tanaka. *"Combining Multiple Evidence from different types of Thesaurus for Query Expansion",* Annual ACM Conference on Research and Development in Information Retrieval (Pág: 191-197), 1999.

[Mitr98] Mandar Mitra – Amit Singhal – Chris Buckley. *"Improving Automatic Query Expansion",* Annual ACM Conference on Research and Development in Information Retrieval (Pág: 206-214),

1998.

[VanR79] C. J. Van Rijsbergen. *"Information Retrieval",* London: Butterworths, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[VanR98] C.J. van Rijsbergen - F. Crestani, *"A study of probability kinematics in information retrieval"*, ACM Transactions on Information Systems 16(3) (Pág: 225-255), 1998.

[Voor94] Ellen M. Voorhees. *"Query Expansion using Lexical-Semantic Relations",* Annual ACM Conference on Research and Development in Information Retrieval (Pág: 61-69), 1994.

[Xu96] Jinxy Xu – W. Bruce Croft, *"Quey Expansion Using Local and Global Document Analysis"*, Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996.

APÉNDICE A

Algoritmos

En este apéndice se explican los algoritmos más importantes que fueron implementados en la aplicación. Algunos de ellos fueron analizados profundamente y se les hace un gran hincapié. Esto se debe a que resultaban críticos para el correcto desempeño de la aplicación.

A.1 Consulta en el directorio de categorías

A continuación se detalla como se implementó la búsqueda de información relacionada con los términos ingresados por el usuario en la base de datos.

A.1.1 Categorías relacionadas

Inicialmente, se genera una colección de palabras a partir de la consulta introducida por el usuario.

Entonces, para cada palabra:

- Se obtiene su lema.
- Se genera una consulta a la base de datos, requiriendo columnas de la tabla **words** (ver definición de tablas en la sección 3.10.1) cuyo campo **word** coincida con el lema obtenido.
- En caso de haber resultados, estos son retornados como una cadena de caracteres separadas por "|".
    - Se genera una colección obteniendo las palabras separadas por "|", cada elemento de la colección es un camino de la ontología *ODP*.
    - Se retorna esa colección.
- En otro caso
    - Se retorna una colección vacía.

A continuación, se eliminan resultados repetidos, que puedan haber surgido de camino relacionados con más de un término de los ingresados a la consulta.

Enseguida, se desarman los caminos obtenidos para evitar la transmisión de información redundante.

Esta transformación es realizada de la siguiente manera:

Se genera una colección vacía de elementos a transmitir.
Para cada categoría a transmitir:

- Si no fue agregada a la colección
    - Se agrega una cadena de caracteres con el formato especificado en la sección 3.8.1.2
- En caso contrario
    - Se agrega una marca, en la cadena de la colección, con referencia al término del cual provino el resultado.

Por ejemplo, si como resultado de haber buscado la palabra "*hard*", se desea transmitir:

```
Top/Arts/Movies/Hard_To_Kill
Top/Arts/Movies/Die_Hard
Top/Arts/Movies/Hard_Days
```

Figura B.0.1 Formato del documento *XML* a transmitir antes su compresión

Estas cadenas se transforman a:

```
Top:0:0
Arts:1:0
Movies:2:1
Hard_To_Kill:3:2
Die_Hard:4:2
Hard_Days:5:2
```

Figura B.0.2 Formato modificado

Y por último se agrega un identificador que marque la palabra de la consulta de la cual provino el resultado. Como en este caso la consulta consta de una única palabra, entonces todos los resultados tendrán el identificador 0 agregado al final.

Quedando como mensaje a transmitir el siguiente:

```
Top:0:0+0
Arts:1:0+0
Movies:2:1+0
Hard_To_Kill:3:2+0
Die_Hard:4:2+0
Hard_Days:5:2+0
```

Figura B.0.3 Datos a transmitir

Nota: En este caso fue utilizado el carácter "+" para separar la relación entre la cadena con información de la categoría y la marca con información de la consulta.

Por último, se genera un solo mensaje a partir de esta colección de resultados, para lograr una transmisión más eficiente por el canal de datos.

A.1.2 Subcategorías

A partir de una categoría de la ontología *ODP*, se deben obtener todas las categorías que la tienen como ancestro directo ("padre").

Esto se implementa de la siguiente manera:

- Se utiliza la tabla **subtopics** (ver definición de tablas en la sección 3.10.1) para obtener el identificador de la categoría a partir del camino (*path*) especificado por el cliente.
- Una vez obtenido el identificador, se vuelve a buscar en **subtopics** todas las categorías que la tienen como ancestro.
- En este caso, se envía la información al cliente como una colección de categorías. Se decidió resolverlo de esta manera, y no como la obtención de categorías relacionadas (ítem anterior) debido a que la cantidad de resultados a retornar nunca es demasiado grande.

A.2 Obtención de sugerencias

A continuación se detalla como, a partir de la información obtenida de la interacción con el usuario, se buscan sugerencias para ofrecer como nuevos términos a agregar en la expansión de la consulta.

El siguiente algoritmo es el más complicado de los implementados en el sistema. Este se debe a la cantidad de parámetros a manejar para tomar decisiones y a la complejidad del problema en cuestión.

Se decidió no tomar en cuenta relaciones semánticas a la hora de resolver el problema ya que no había tiempo para realizar esta clase de estudios. Sin embargo, el algoritmo propuesto tiene en cuenta información estadística que en una gran cantidad de situaciones logra suplir la falta de información semántica.

La información utilizada para ofrecer sugerencias está dada por:

- Consulta ingresada inicialmente por el usuario.
- Categorías seleccionadas de la ontología.

El método es el siguiente:

1. Se genera una colección de categorías utilizando las seleccionadas por el usuario y todos los ancestros de cada una de ellas.
2. Se obtiene una colección de palabras a partir de la consulta inicial.
3. Para cada palabra:
    a. Se obtiene el conjunto de significados que tiene.
    b. Para cada significado:
        i. Se obtiene recursivamente una cantidad fijada de niveles de hiperonimia. Es decir, se consiguen los hiperónimos para un sentido, luego los hiperónimos de esos, y así hasta que se llega al nivel deseado. Esta información está determinada por la constante **hipernymLevels**, que para esta implementación tiene asignada el valor 4.
        ii. Se genera una lista de hiperónimos, colocando más adelante los que están más "cerca" del sentido real de la palabra.
        iii. Se retorna una colección de listas de significados.
    c. Para cada elemento de la colección:
        i. Se genera una lista de palabras (obtenidas de las definiciones) ponderadas de la siguiente manera. Una palabra es más importante (tiene mayor valor) si está más "cerca" del sentido de la palabra. Aquí se pueden tomar dos decisiones, una es tener en cuenta la repetición de términos y la otra es no tenerlos en cuenta. Se podría pensar que una palabra es más relevante en la definición del sentido de un término si aparece muchas veces, sin embargo, las pruebas realizadas muestran que esto no es correcto (o por lo menos no tan natural como parece a simple vista), por lo tanto se decidió no tener en cuenta las repeticiones de términos en las definiciones. Igualmente esta decisión puede ser modificada ya que esta determinada por la constante **countRepeated** que en este caso tiene el valor 0.
        ii. Se compara la lista de palabras con la colección de categorías obtenida en el paso nº 1 de la siguiente manera:
        iii. Se inicia un contador en 0 y se suma el valor de relevancia de una palabra cada vez que es encontrada en ambos conjuntos.
    d. Con esto se obtiene para cada sentido:
        i. La cantidad de palabras que lo definen.
        ii. Un valor que marca las coincidencias entre las palabras del sentido y las del conjunto de categorías.
    e. Con esta información se procede a elegir el sentido más relacionado con el término ingresado por el usuario. Esto se realiza comparando los valores de concordancia

obtenidos para cada sentido. En caso de que los valores de concordancia sean muy bajos, se comparan con un mínimo marcado para eliminar valores muy pequeños de relevancia. Este dato está dado por la constante **tresholdQuery** que tiene el valor 1. Puede suceder que la información de concordancia sea la misma para dos o más sentidos. En esta situación se elige el primero encontrado, lo que en *WordNet* es lo mismo que decir: el más probable, ya que están ordenados por la probabilidad de aparición en un texto cualquiera. Sin embargo, existe una situación en la que está opción no es tenida en cuenta: es cuando los valores de concordancia son iguales para un sentido con la categoría sustantivo y otro con una categoría diferente. Únicamente para este caso, se tomó la decisión de elegir la definición que hace relación al sustantivo. Esta disposición viene acompañada del análisis de la manera en que uno busca información, por lo general, uno ingresa sustantivos en una consulta.

f. Una vez obtenido el sentido más relacionado, se le ofrece este al usuario, junto con la lista de palabras que referencian a ese sentido, ordenadas por su valor de relevancia.

A.3 Generación de una consulta

La consulta se genera para motores de búsqueda que soportan el operador booleano *or*. En caso de desear generar una consulta para un motor que no cumpla esa condición, habría que modificar el algoritmo. No alcanza con eliminar el operador de la consulta porque en ese caso se perdería una gran cantidad de resultados relevantes.

El programa está limitado a generar una consulta de, como máximo, 10 palabras, ya que el buscador principal utilizado para este proyecto, o sea *Google*, impone esa limitación al largo de sus consultas. Igualmente este valor podría ser modificado ya que está definido por la constante **maxWords**.

La información utilizada para generar esta consulta surge de la interacción con el usuario. Específicamente se está usando:

- Consulta ingresada inicialmente por el usuario.
- Categorías seleccionadas de la ontología *ODP*.
- Sugerencias elegidas a partir de los sentidos ofrecidos por el sistema.

La metodología seguida es la siguiente:

1. Primeramente, se filtran las palabras ingresada por el usuario y las categorías elegidas, eliminando las llamadas *Stop Words* o palabras que no sirven de ayuda.

2. Para cada palabra ingresada (luego de filtradas):
    a. Si existen sugerencias elegidas para esa palabra:
        i. Se agrega cada una de ellas utilizando el operador *or*.
    b. Estas son agregadas a la consulta en el orden en que fueron escritas.

3. Para cada categoría elegida:
    a. Se obtienen todos los sentidos posibles para esa categoría.
    b. Se utiliza la información detallada a continuación para elegir un sentido apropiado para la definición de la categoría:
        - las palabras ingresadas
        - las restantes categorías seleccionadas
        - las sugerencias elegidas
    c. Con esta información se intenta elegir un sentido en forma similar a lo mostrado en el ítem anterior. Sin embargo, para tomar la decisión de elegir una definición, se utiliza un valor de concordancia mínimo mayor que el elegido anteriormente. En este caso se utiliza la constante **thresholdCategory** para marcar este mínimo (definida con valor 2 en el código implementado).
    d. En caso de lograr obtener un sentido:

i. Se utiliza el sinónimo más probable para expandir la consulta, uniendo la categoría con el operador *or*.

e. En caso contrario:

i. Se agrega a la consulta únicamente la categoría.

A.4 Generación de un perfil de usuario

El usuario tiene la posibilidad de especificar sus áreas de interés. Esto será útil en el momento de obtener resultados de la ontología ya que el grafo dibujado tendrá marcadas de un color especial las aristas que llevan a categorías relacionadas con sus intereses.

Para crear un perfil de usuario se debe abrir la ventana para este propósito. Una vez abierta, se debe especificar el nombre que se le desea dar al nuevo perfil. De lo contrario, se puede tomar uno ya creado y modificarlo.

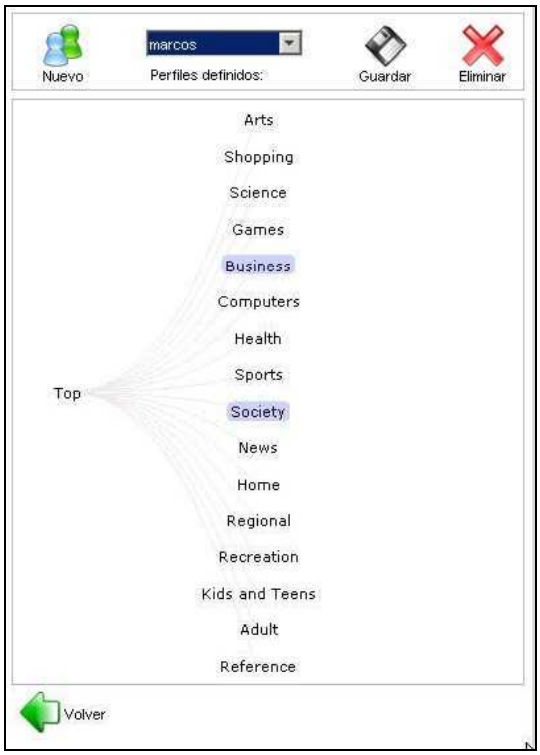

Figura 0.4 Ventana para manejar perfiles de usuario

Enseguida se obtienen todas las categorías de nivel 2 de la ontología. Esto se realiza a través de una consulta con el servidor. Los resultados se muestran en un árbol similar al que se despliega al mostrar categorías relacionadas.

A continuación se permite al usuario elegir las categorías que considere de su interés. Incluso puede elegir subcategorías de estas, o sea, categorías de nivel 3.

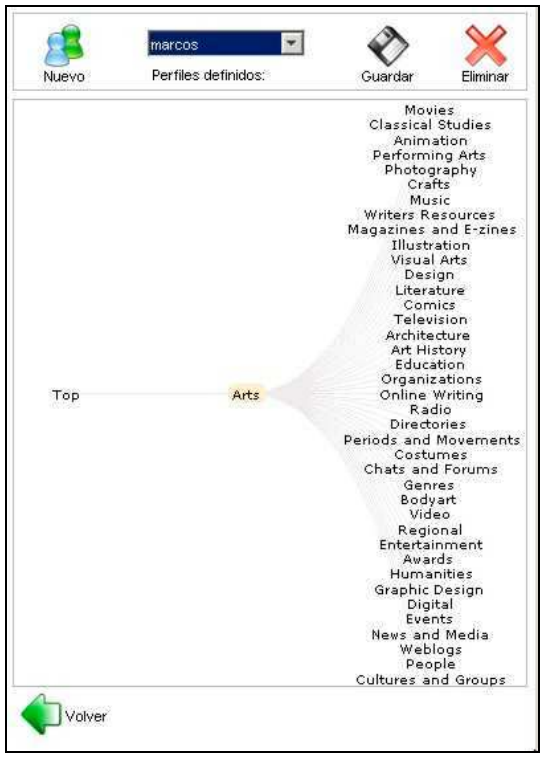

Figura 0.5 Selección de categorías más específicas

Una vez elegidas las categorías deseadas, este puede guardar el perfil.

Toda la información obtenida de esa interacción es guardada en el cliente, en un archivo de configuración. No queda ninguna información en el servidor sobre el o los perfiles locales de cada usuario. Esto no pareció importante porque en definitiva, las preferencias del usuario se utilizan únicamente para marcar de manera diferente las categorías obtenidas como resultado.

APÉNDICE B

B

Interfaz Gráfica

B.1 Manejo del árbol de resultados

Cuando el usuario realiza una consulta, la aplicación cliente visualiza el conjunto de resultados mediante un árbol navegable. Sin lugar a dudas, dicho árbol de resultados es el componente más complejo de la aplicación cliente.

Como se dijo anteriormente, el desarrollo del mismo está basado en la biblioteca *Prefuse* (ver sección 4.6: Tecnologías utilizadas). En esta sección se explican los pormenores del funcionamiento del árbol de resultados.

Toda consulta realizada al servidor retorna un conjunto de categorías. Esas categorías conforman una jerarquía con forma de árbol. Dicho árbol puede contener una gran cantidad de nodos, por lo que algunas veces resulta imposible visualizarlo completamente en la pantalla del usuario. Se brindan entonces funcionalidades que permiten mostrar las regiones del árbol que el usuario considere relevantes, sin que la cantidad de información mostrada en cada momento imposibilite su visualización.

Las funcionalidades implementadas consisten en permitir que el usuario navegue por el árbol de resultados. Dicha navegación ocasiona que nuevas regiones del árbol sean mostradas, mientras las demás son ocultadas. Para evitar que el usuario se sienta abrumado por la cantidad de resultados obtenidos, también se provee de un mecanismo de filtrado de categorías en tiempo real. Estas y otra funcionalidades son las que se pasa a describir a continuación.

B.1.1 Navegación del árbol de resultados

Inicialmente, el árbol de resultados muestra la categoría raíz (la cual es denominada siempre como "*Top*") y las categorías que surgen de ella hasta una cantidad determinada de subniveles. La cantidad de subniveles que se muestran en cada momento (profundidad de la visualización) puede ser controlada por el usuario (siendo cuatro la máxima profundidad y uno la mínima).

La navegación en el árbol es manejada seleccionando (haciendo *click* con el botón izquierdo del ratón) las categorías (o nodos) del árbol. En la Figura B.0.1 se muestran los resultados iniciales para la consulta "*jaguar*" y en la Figura B.0.2 lo que se puede observar luego de navegar a través de las categorías "*Science*" y "*Biology*".

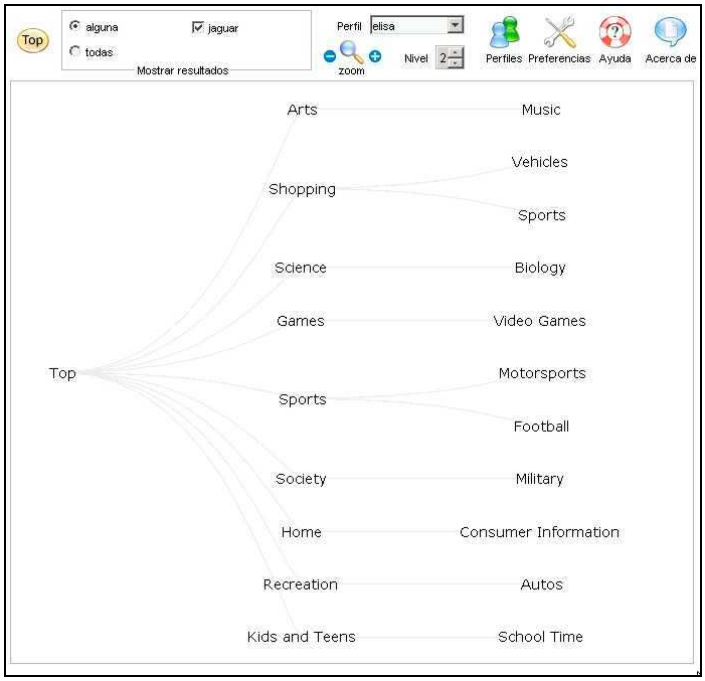

Figura B.0.1 Resultados iniciales

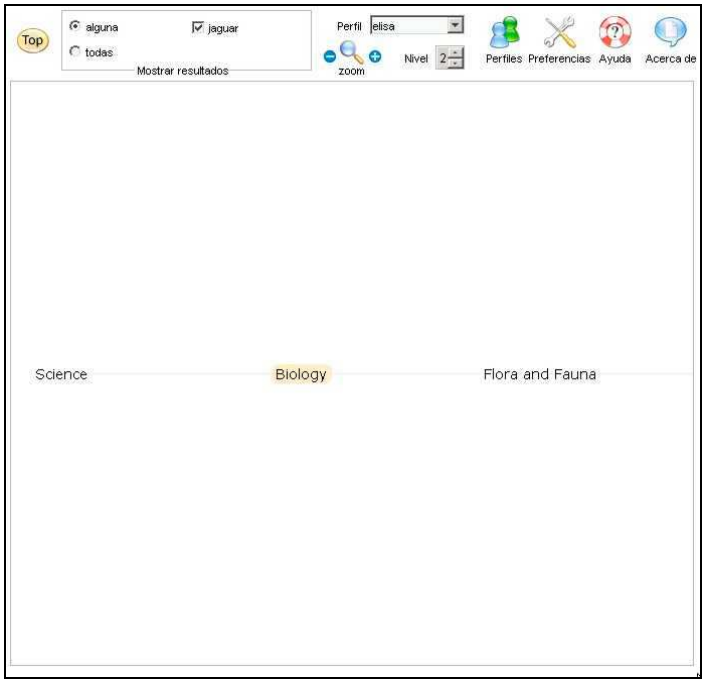

Figura B.0.2 Navegación de resultados

Cuando el usuario hace *click* sobre una categoría del árbol ocurre lo siguiente:

- El ancestro del nodo (categoría más general) sobre el que se hizo clic se convierte en la nueva raíz de árbol.
- Se oculta el ancestro y los hermanos de la nueva raíz.
- Se ocultan todos los hermanos del nodo seleccionado.
- Se visualizan los descendientes del nodo elegido, hasta la profundidad que el usuario tenga configurada en ese momento.

El mecanismo descrito permite entonces descender por la jerarquía. Cuando el usuario desea volver hacia atrás (ascender por la jerarquía), debe seleccionar la raíz del árbol (en ese momento). Esto ocasiona que:

- El padre de la raíz actual se convierta en la nueva raíz.
- Se visualicen todos los descendientes de la antigua raíz hasta la profundidad configurada en ese momento.

La interfaz de usuario brinda también un botón (con la etiqueta "*Top*") que ocasiona que el árbol de resultados vuelva a su forma inicial (la que tenía inmediatamente después de realizar la búsqueda).

Seleccionando con el botón derecho del ratón sobre una categoría, se despliega una lista conteniendo todas las categorías hoja (aquellas sin descendientes) que son descendientes de la categoría elegida. Las categorías hoja son aquellas que están directamente relacionadas con alguna de las palabras que el usuario buscó. Al marcar sobre alguna de las categorías de esta lista se navega directamente hacia ella. Este mecanismo evita que la navegación se haga tediosa, ya que se saltan muchos niveles con un solo *click*.

B.1.2 Filtrado de categorías en el árbol de resultados

Para evitar que el usuario se vea desbordado por la cantidad de resultados mostrados simultáneamente, se desarrolló, además de la unidad de navegación, un mecanismo de filtrado de resultados en tiempo real.

Para que una categoría aparezca en el árbol de resultados, esta debe estar relacionada con al menos una de las palabras que el usuario ingresó. Si este buscó tres palabras, una categoría del árbol puede estar relacionada con una de las palabras, con dos o con todas. Supóngase entonces que luego de realizar la búsqueda, la cantidad de resultados desplegados es tal que el usuario se ve confundido. El mecanismo de filtrado le permite entonces disminuir la cantidad de resultados mostrados, solicitando que se visualicen únicamente las categorías relacionadas con todas o algunas (se le permite seleccionar cuales) de las palabras buscadas.

Las figuras Figura B.0.3, Figura B.0.4 y Figura B.0.5 permiten observar estas opciones.

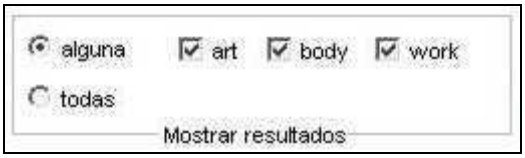

Figura B.0.3 Mostrar relacionadas con "art" o "body" o "work"

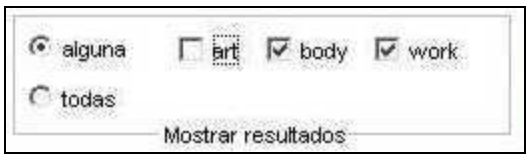

Figura B.0.4 Mostrar relacionadas con "body" o "work"

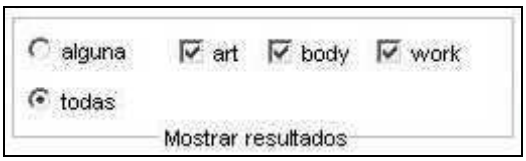

Figura B.0.5 Mostrar relacionadas con "art" y "body" y "work"

B.1.3 Resaltado de caminos

Como se indicó anteriormente cada categoría de las que se encuentran presentes en el árbol está relacionada con al menos una de las palabras buscadas. Dependiendo de la cantidad de palabras con las que una categoría esté relacionada, se determina el grosor del camino que llega hasta la misma desde su ancestro (a más palabras relacionadas mayor grosor). Este permite que a un golpe de vista el usuario determine las categorías más relevantes para su búsqueda.

También se utiliza la información del perfil de usuario utilizado para diferenciar categorías. Como se puede observar en la Figura B.0.6, los caminos que llegan hasta las categorías "*Business*" y "*Society*" están marcados con otro color, esto indica que esas categorías pertenecen al perfil que el usuario definió y por lo tanto, es probable que sean de mayor interés.

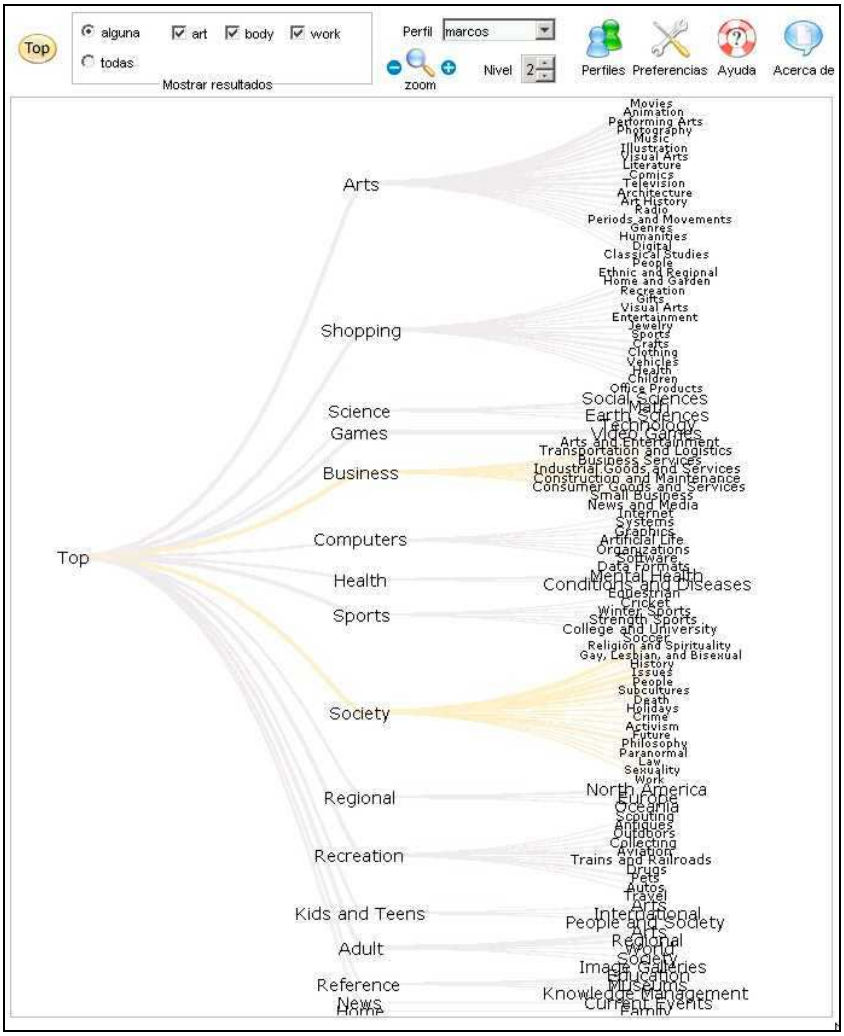

Figura B.0.6 Caminos resaltados por coincidencia con perfil de usuario

B.1.4 Organización del árbol

El conjunto de categorías visibles es dinámico por lo cual es necesario calcular a cada momento la distribución en la pantalla de las mismas. Dicha distribución es estéticamente agradable, clara y consistente con la topología de los resultados (la forma es siempre arborescente).

Los nodos que se encuentran a igual profundidad aparecen en la misma coordenada en el eje *X*. La coordenada en el eje *Y* se calcula en función de la cantidad de descendientes visibles y la cantidad de hermanos de cada nodo.

B.1.5 Opciones de zoom

Ante las diferentes cantidades de resultados o mostrar, la aplicación provee la posibilidad de acercar y alejar la imagen para poder visualizar correctamente el volumen de información obtenido.

En la Figura B.0.7 se puede ver el resultado de acercar la imagen pudiendo visualizar más claramente las descripciones de las categorías, y en la Figura B.0.8 se muestra un alejamiento de la imagen obteniendo una visión global de la estructura obtenida.

Figura B.0.7 Acercamiento de la imagen

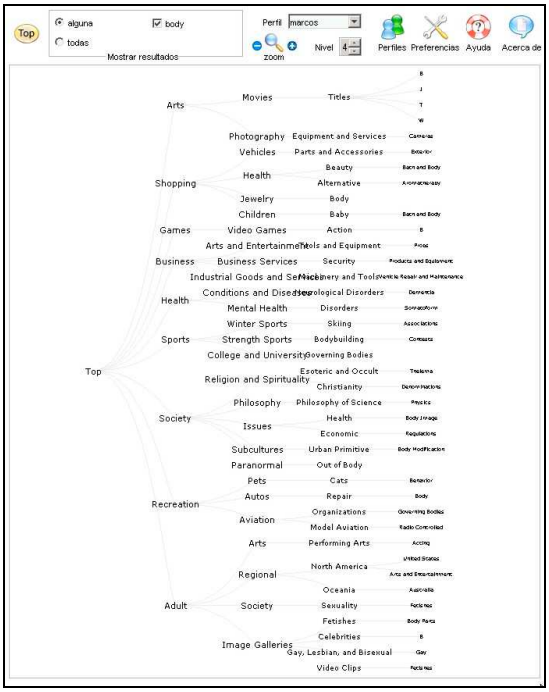

Figura B.0.8 Alejamiento de la imagen

B.1.6 Cantidad de niveles

Otra manera de distinguir los resultados obtenidos es modificar la cantidad de niveles que se desean mostrar. La Figura B.0.9 muestra el árbol de resultados desplegando un único nivel de hijos para la categoría "*Top*". La Figura B.0.10 muestra el mismo conjunto con cuatro niveles en la jerarquía.

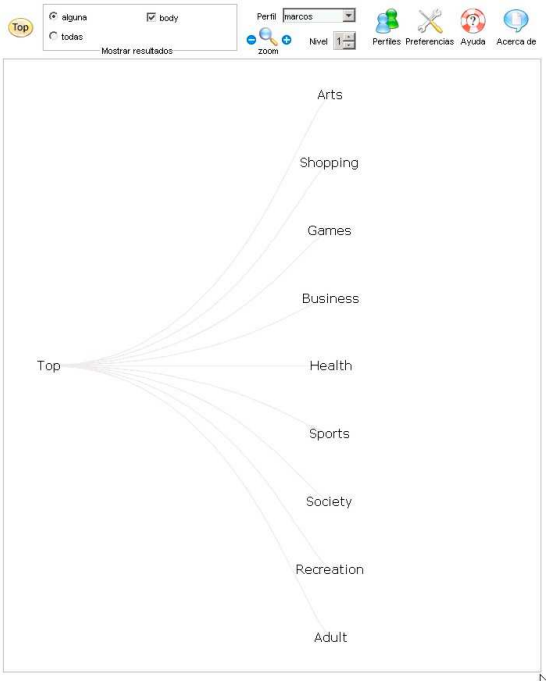

Figura B.0.9 Visión de un nivel

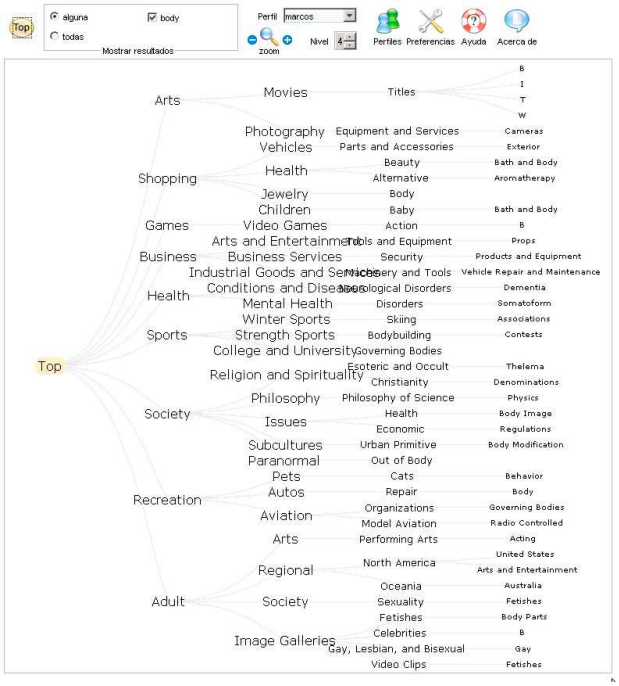

Figura B.0.10 Visión de cuatro niveles

B.2 Ayuda

Al desarrollar una aplicación para usuarios con poco conocimiento sobre *Recuperación de Información* resulta importante que la ayuda sea clara y explique en pocos pasos como comenzar a utilizar el sistema y obtener resultados provechosos rápidamente. Sin embargo, también es importante que la ayuda explique cada una de las funcionalidades de la aplicación lo cual puede resultar contradictorio con la necesidad anterior.

Para tener en cuenta todos estos puntos se decidió realizar dos instancias de ayuda con características diferentes:

- Ayuda en línea: introduce a la utilización de la aplicación dando la serie de pasos necesaria para generar una consulta. Esta se muestra mientras se está ejecutando la aplicación.
- Ayuda extendida: se muestra a través de una ventana del navegador que posea el sistema. Contiene información detallada de cada una de las utilidades de la aplicación y muestra imágenes explicativas de las ventanas de esta. Está generada en *HTML* para su fácil acceso.

B.2.1 Ayuda en línea

Como ya se explicó, proporciona una secuencia de pasos a seguir para utilizar la aplicación. Está diseñada como una ventana aparte de las dos centrales que conforman el sistema. Consta de un cuadro con texto donde se explica el próximo paso a dar para generar una consulta y de botones para avanzar y retroceder en la secuencia.

La ventana permite ser ocultada cuando el usuario se considere lo suficientemente ágil en el sistema como para no utilizarla.

Cuándo se inicia la ayuda se puede ver la imagen de la Figura B.0.11.

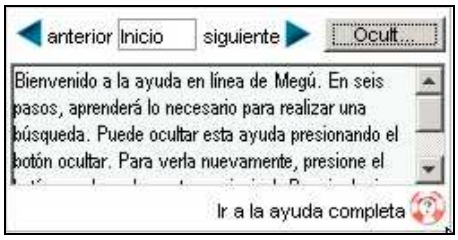

Figura B.0.11 Inicio de la ayuda

Presionando el botón con la etiqueta "siguiente" se avanza hacia el primer paso para utilizar la aplicación.

La secuencia de eventos especificada en la ayuda para generar una consulta es la siguiente:

- Ingresar las palabras a buscar (Figura B.0.12).
- Presionar el botón con la etiqueta "buscar" (Figura B.0.13).
- Seleccionar las categorías deseadas del árbol de resultados (Figura B.0.14).
- Presionar botón "sugerencias" para obtener la desambiguación de términos y las palabras relacionadas (Figura B.0.15).
- Elegir palabras relacionadas (Figura B.0.16).
- Presionar botón "*Google*" para generar la consulta y enviarla al motor de búsqueda (Figura B.0.17).

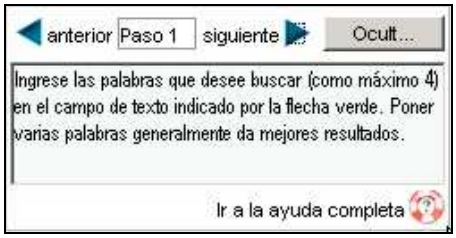

Figura B.0.12 Paso Nº 1

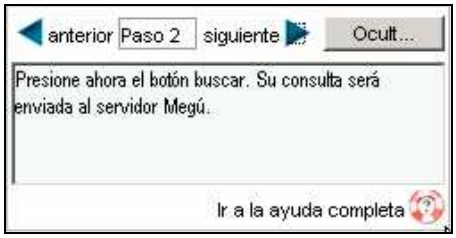

Figura B.0.13 Paso Nº 2

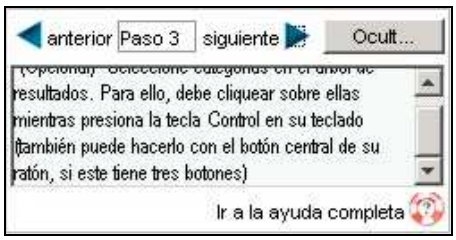

Figura B.0.14 Paso Nº 3

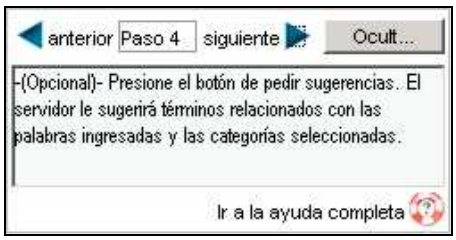

Figura B.0.15 Paso Nº 4

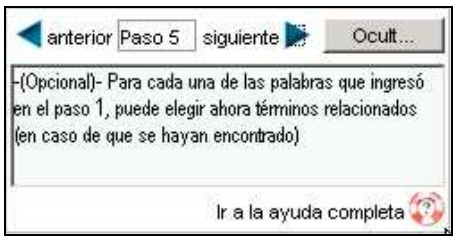

Figura B.0.16 Paso Nº 5

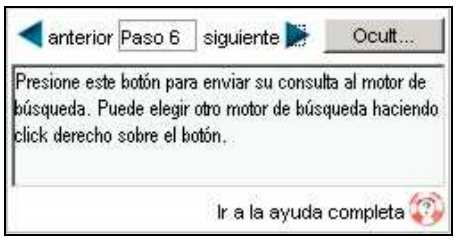

Figura B.0.17 Paso Nº 6

B.2.2 Ayuda extendida

Esta ayuda no está orientada a la generación de una consulta sino a la explicación detallada y pormenorizada de cada una de las funcionalidades que ofrece la aplicación cliente. Está escrita en *HTML* para su fácil acceso y lectura. Consta de un documento específico para la explicación de las opciones de cada ventana. Además se puede seleccionar (hacer *click* con el botón izquierdo del ratón) sobre la información de las imágenes mostradas para obtener comentarios al respecto.

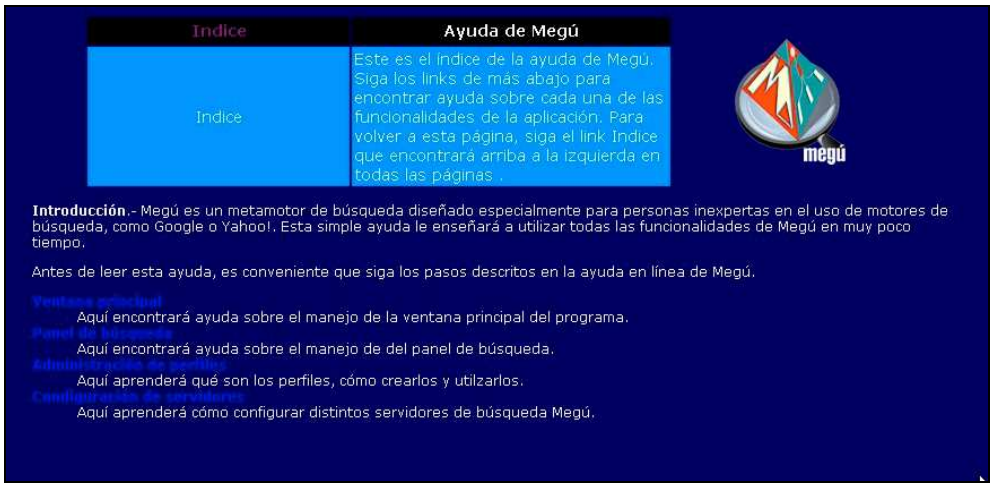

Figura B.0.18 Inicio de la ayuda extendida

La Figura B.0.18 muestra la página principal de esta ayuda. Contiene el índice y los vínculos a la información referente a cada ventana. Estos vínculos son:

- Ventana principal.
- Panel de búsqueda.
- Administración de perfiles.
- Configuración de servidores.

Ventana principal:

La página especifica sus funcionalidades. Como se puede ver en la Figura B.0.19, tiene marcada cada función y seleccionando cada una de estas se puede obtener una explicación concreta de su utilidad. Las opciones ofrecidas son:

- Botón de ir a "*Top*".
- Controles de filtrado.
- Controles de *zoom*.
- Selector de perfiles.
- Selector de nivel.
- Configuración de perfiles.
- Configuración del programa.
- Botón de ayuda.
- Botón para obtener información de la aplicación.

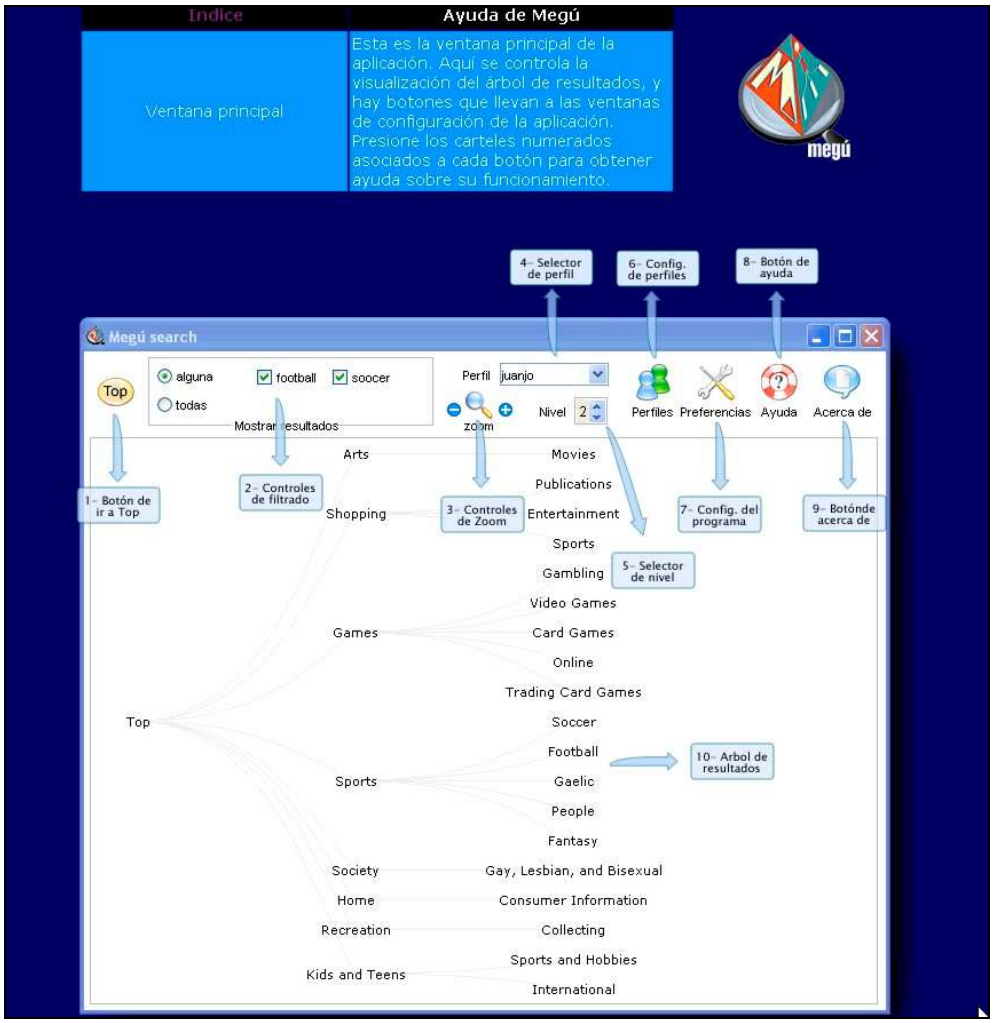

Figura B.0.19 Ventana principal

Panel de búsqueda:

Aquí se puede obtener una explicación de las utilidades de sus botones y campos a rellenar (Figura B.0.20). Las opciones explicadas son:

- Campo de búsqueda.
- Botón de búsqueda.
- Lista de categorías.
- Botón de sugerencias.
- Lista de sugerencias.
- Botón de generación de la consulta.

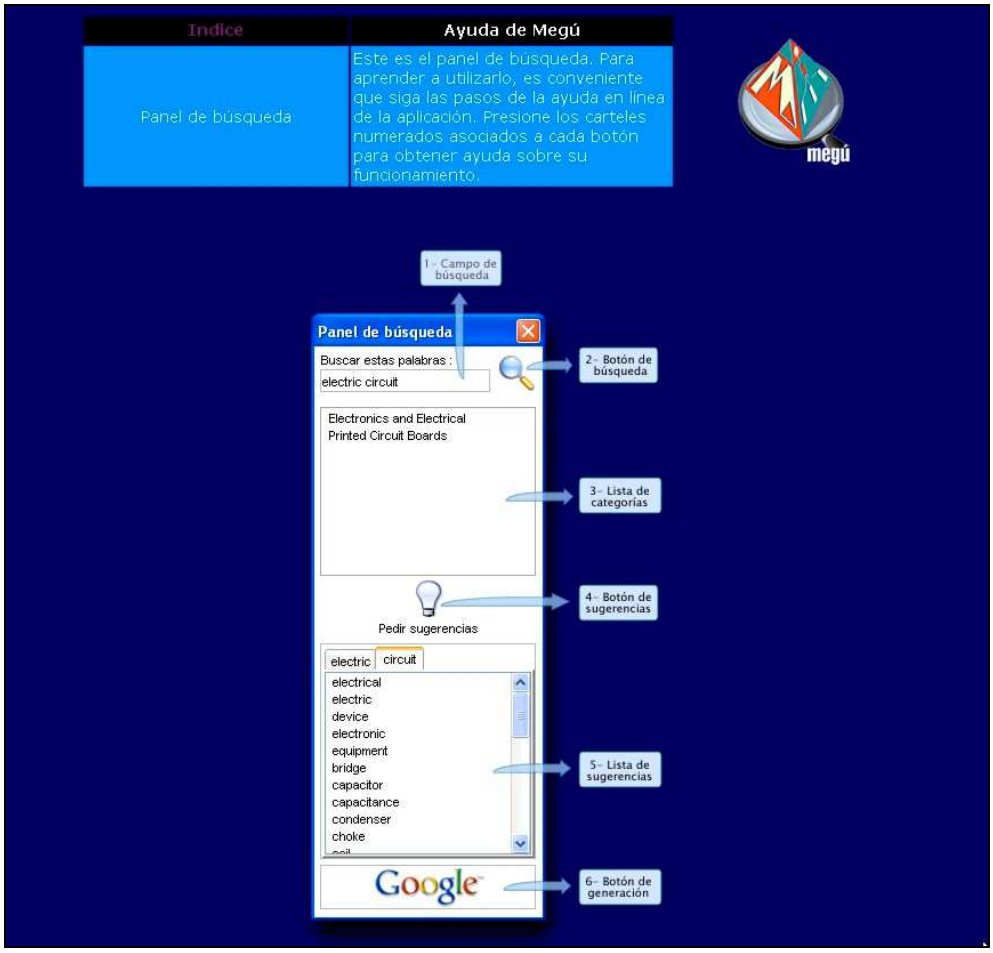

Figura B.0.20 Panel de búsqueda

Administración de perfiles:

La administración del perfil (Figura B.0.21) permite generar, cambiar y borrar perfiles de usuario. Las opciones manejadas son:

- Nuevo perfil.
- Selector de perfil.
- Guardar cambios.
- Eliminar perfil.
- Árbol de categorías.
- Botón para volver a la aplicación.

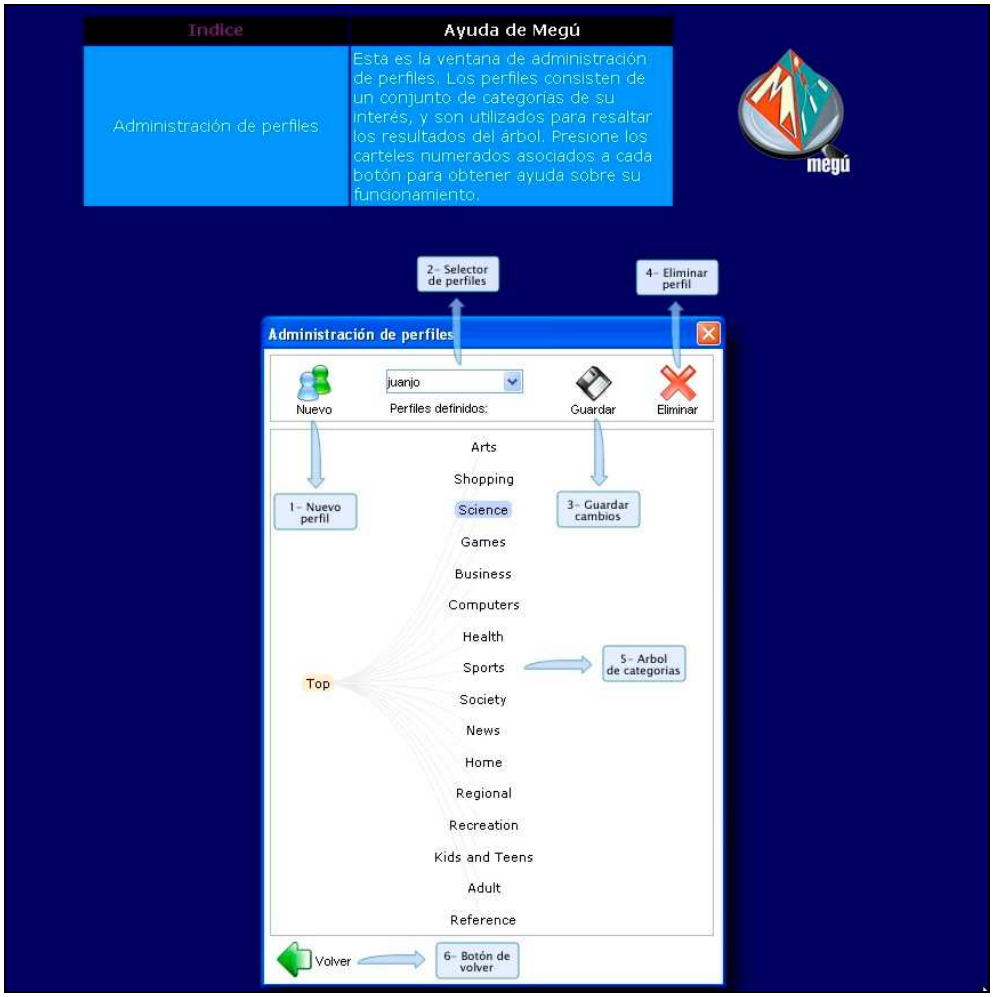

Figura B.0.21 Administración de perfiles

Configuración de servidores:

Desde aquí (Figura B.0.22) se puede especificar la información para conectarse a un servidor de *megú*. Se especifican las siguientes alternativas:

- Agregar un servidor.
- Utilizar un servidor.
- Borrar un servidor.
- Verificar la conexión con un servidor.
- Servidor en uso.
- Lista de servidores.
- Botón para volver a la aplicación.

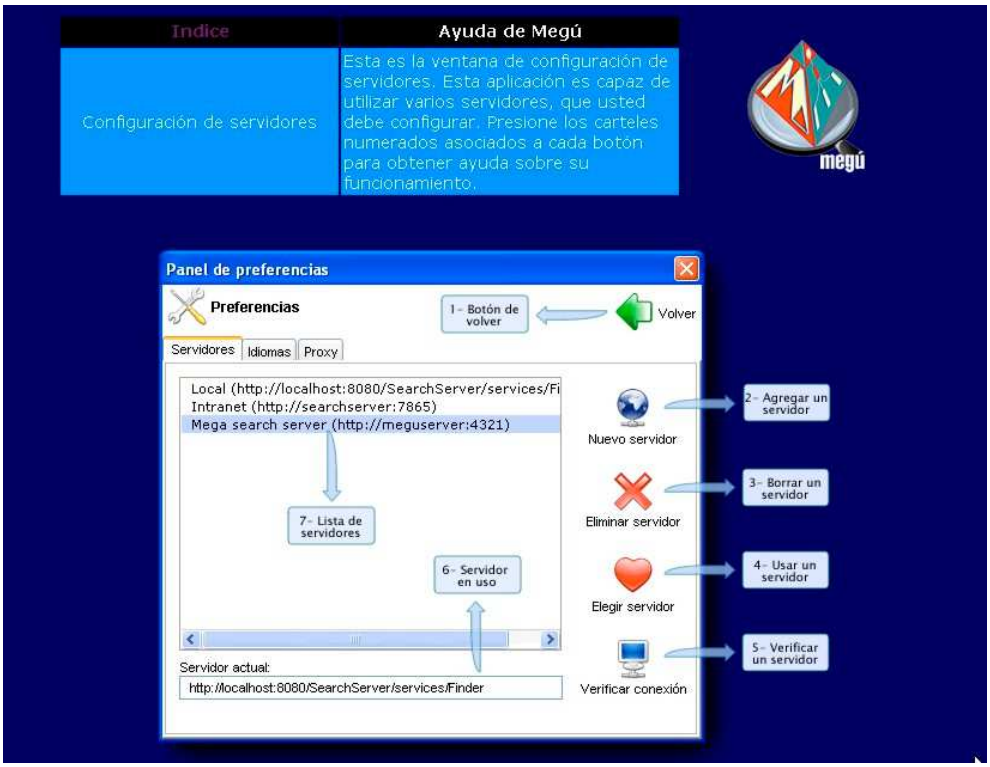

Figura B.0.22 Configuración de servidores

APÉNDICE C

C

Configuración del Servidor

El servidor fue desarrollado como una aplicación sin necesidad de ser controlada. Únicamente se debe levantar el sistema y dejarlo funcionando, para que este realice su tarea sin problemas. Sin embargo, si fue especificado un archivo de configuración para determinar algunas variables que se establecen en el proceso de instalación del sistema.

Este archivo de configuración consta de 5 secciones:

- *JWNL section.*
- *Database section.*
- *Server section.*
- *Stopwords section.*
- *Suggestions section.*

JWNL section:

Esta sección permite establecer el siguiente parámetro:

- *JWNLInitFile.*

Este es el archivo de especificación de propiedades de la *JWNL* (sección 4.6.1: Tecnologías utilizadas en el servidor).

Database section:

Está dedicada a la configuración de la base de datos y contiene los siguientes parámetros:

- *dBConnectionClass*: nombre de la clase que implementa la interfaz *IDBConnection* (sección 4.7: Diseño del servidor).
- *databaseName*: nombre de la base de datos.
- *databasePort*: puerto de conexión a la base de datos.
- *databaseUser:* nombre de usuario para conectarse a la base de datos.
- *databasePassword:* contraseña para ese nombre de usuario.

Server section:

Generada para especificar los mensajes que da el servidor cuando está funcionando correctamente. Esta consta únicamente del parámetro:

- *serverResponse:* que establece el mensaje que le indica al cliente cuando este está disponible.

StopWords section:

Esta consta de un único parámetro:

- *stopWordsFile.*

Dicho parámetro especifica donde se encuentra el archivo que determina cuáles palabras no deben ser tenidas en cuenta por el servidor cuando el cliente se las envíe a través de una consulta.

Este parámetro debe tener la dirección dentro del sistema de ficheros de un archivo de texto que contenga una lista de palabras separados por el carácter de fin de línea del sistema.

Suggestions section:

La sección tiene como cometido especificar ciertos parámetros que permiten modificar en algún sentido el comportamiento del sistema. Esta consta de 5 parámetros:

- *hypernymLevels*: permite definir la cantidad de hiperónimos que se desea agregar en el conjunto de términos relacionados con la palabra a desambiguar. El sistema desarrollado tiene asignado un 4 a este parámetro.
- *maxRelationships*: es el valor máximo de relevancia que se le da a un término relacionada con la palabra a desambiguar. A términos menos relevantes se les dará un valor menor que este. El sistema desarrollado tiene asignado un 20 a este parámetro.
- *thresholdQuery=1*: especifica la mínima diferencia de valores de concordancia entre sentidos de las palabras (ingresadas inicialmente) para considerar cada una de ellas desambiguada. Si la diferencia entre sentidos es menor que este valor, entonces el término no puede ser desambiguado. El sistema desarrollado tiene asignado un 1 a este parámetro.
- *thresholdCategory=2*: especifica la mínima diferencia de valores de concordancia entre sentidos de las categorías (elegidas de la ontología) para considerar cada una de ellas desambiguada. Si la diferencia entre sentidos es menor que este valor, entonces la categoría no puede ser desambiguada. El sistema desarrollado tiene asignado un 2 a este parámetro.
- *allSynonymsWithOr=1*: determina como agregar los términos relacionados elegidos del conjunto de sugerencias. Los términos que son sinónimos se agregan siempre con el operador *or*, pero los que no lo son (hiperónimos, hipónimos, merónimos, etc) pueden agregarse con *or* o con *and*. Si se especifica este parámetro con el valor 1, entonces son agregados con el operador *or*, en caso contrario, con *and*. El sistema desarrollado tiene asignado un 1 a este parámetro.

El archivo de configuración está dividido en secciones por medio de comentarios. Dicha división existe solo para aumentar la claridad del mismo y no es necesaria para el funcionamiento del *ServerConfigManager* que es la clase que interactúa con este archivo.

A continuación se muestra un ejemplo de este archivo:

```
#Server config file

#JWNL section

JWNLInitFile=F:/jwn/file_properties.xml
(ubicación en el sistema del archivo que contiene los parámetros de inicialización del diccionario WordNet)

#database section

dBConnectionClass=db.MySQLDBConnection (nombre de la clase que implementa la interfaz IDBConnection)

databaseName=odpnew (nombre de la base de datos que contiene la ontología)

dataBaseHost=localhost (dirección IP o nombre de la máquina que contiene la base de datos)

dataBasePort=3306 (puerto en el cual atiende conexiones el DBMS)

dataBaseUser=root (usuario a utilizar para acceder a la base)

dataBasePassword=root (password a utilizar para acceder a la base)

#server section

serverResponse=Server OK (respuesta del servidor cuando se le invoca el método getServerInfo() sin parámetros)

#stopwords section
```

`stopWordsFile=C:/stopwords.txt` (ubicación en el sistema del archivo que contiene *stop words*)

`#suggestions section`

`hypernymLevels=4` (niveles que se sube en la jerarquía para obtener términos de comparación)

`maxRelationships=20` (valor máximo que se le asigna a un término relacionado con la palabra a desambiguar)

`thresholdQuery=1` (diferencia en valores de concordancia entre sentidos para considerar una palabra desambiguada, esta se utiliza cuando se desambigua un término de la consulta inicial)

`thresholdCategory=2` (diferencia en valores de concordancia entre sentidos para considerar una palabra desambiguada, esta se utiliza cuando se desambigua un término de los elegidos en las categorías de la ontología)

`allSynonymsWithOr=1` (agregar todos los términos elegidos utilizando el operador *or*)

Figura 0.1 Archivo de configuración del servidor

Como se comentó anteriormente, es posible agregar nuevos parámetros al archivo de configuración. Para accederlos, se debe utilizar la siguiente operación pasando como argumento el nombre del parámetro al cual se desea acceder:

```
getConfigParameter (String parameterName):String
```

APÉNDICE D

D

Importador de Bases de Datos

Para la importación de nuevas ontologías se desarrolló una aplicación capaz de leer un archivo en un formato específico, procesar esta información y generar una base de datos capaz de interactuar con el servidor desarrollado en este trabajo.

El archivo de entrada contiene las categorías a importar. Dicho archivo debe contener un árbol que organice las categorías jerárquicamente.

El ejemplo mostrado a continuación permite ver el formato:

```
Art
Art/Movies
Art/Movies/Titles
Art/Movies/Titles/The_Hours
Art/Movies/Titles/Star_Wars
```

Figura 0.1 Formato del archivo de la ontología

Cada línea del archivo contiene un camino desde la raíz hasta una hoja del árbol. Los nodos de los caminos deben estar separados por el carácter "/", y los espacios deben estar sustituidos por el carácter "_". En este contexto, se define una categoría como un camino desde la raíz a un nodo cualquiera del árbol. Con esta definición, *Art* es una categoría y *Art/Movies/Titles* también. Una categoría es subcategoría de otra siempre que la segunda sea un prefijo de la primera. Por ejemplo, *Art/Movies* es una subcategoría de *Art*, pero *Business* no lo es.

A continuación se especifican los pasos que sigue el programa de importación para poblar la base de datos.

Como se vio en la definición de la base de datos (sección 4.10: Definición de la base de datos), las tablas son las siguientes:

Tabla: **Topics**		
topicId : Valor Natural (int)	**path : Arreglo de caracteres (Varchar)**	topicTitle : Arreglo de Caracteres (Varchar)
Clave	Clave	

Tabla: **subTopics**		
path : Arreglo de Caracteres (Varchar)	**subTopicId : Valor Natural (int)**	topicId : Valor Natural (int)
Clave	Clave	Con Índice

Tabla: **words**	
word : Arreglo de Caracteres (Varchar)	paths : Texto (text)
Clave	

Para poblar estas tablas la aplicación ejerce los siguientes pasos:

- Inicialmente lee el archivo de categorías y lo carga en memoria.
- Para cada categoría:
    - Le asigna el siguiente identificador disponible.
    - Inserta la categoría junto con el identificador asignado en la tabla **topics**.
    - Busca su antecesor en la tabla **topics** (por esto es importante que el archivo de categorías venga ordenado de forma que los antecesores estén especificados antes).
    - Inserta la categoría, el identificador suyo y el de su antecesor en la tabla **subtopics**.
    - Se divide la categoría en palabras.
    - Para cada palabra:
        - Se inserta esta en una tabla de dispersión asignando como clave la palabra y como valor la categoría completa. En caso de que ya existiera la palabra en la tabla se concatena la categoría al final del valor que esta tuviera.
- Se vuelca la información de la tabla de dispersión en la tabla **words**.

El proceso de volcado de la información de la ontología en la base de datos es bastante largo. En el caso del conjunto de categorías de *ODP* tarda varias horas en una computadora moderna. En caso de que el sistema cayera o tuviera que ser dado de baja, la aplicación tiene la posibilidad de recuperarse automáticamente y continuar con el trabajo sin tener que volver a comenzarlo. Para esto se define la constante **saveAt** (definida por defecto en 20.000) que define cada cuantas categorías se vuelca la tabla de dispersión en la base de datos.

La aplicación tiene un archivo de configuración para especificar parámetros con información sobre la base de datos, nombre de las tablas y parámetros del programa.

Este archivo de configuración consta de 5 secciones:

- *Source section.*
- *StopWords section.*
- *Database section.*
- *Import Parameters section.*
- *JWNL section.*

Source section:

Esta sección permite establecer el siguiente parámetro:

- *baseFileName.*

Este es la dirección del archivo de categorías con el formato indicado anteriormente

StopWords section:

Esta consta de un único parámetro:

- *stopWordsFile.*

Dicho parámetro especifica donde se encuentra el archivo que determina cuáles palabras no deben ser tenidas en cuenta por el servidor cuando el cliente se las envíe a través de una consulta.

Este parámetro debe tener la dirección dentro del sistema de ficheros de un archivo de texto que contenga una lista de palabras separados por el carácter de fin de línea del sistema.

Database section:

Está dedicada a la configuración de la base de datos y contiene los siguientes parámetros:

- *databaseName*: nombre de la base de datos.

- *databasePort*: puerto de conexión a la base de datos.
- *databaseUser:* nombre de usuario para conectarse a la base de datos.
- *databasePassword:* contraseña para ese nombre de usuario.

Import Parameters section:

Generada para definir variables internas del programa. Esta consta de los parámetros:

- *maxWords:* establece la máxima cantidad de palabras que serán guardadas en la tabla **words**.
- *initialSize*: para evitar el problema de orden de ejecución que existe con la función de concatenación de cadenas de caracteres que provee *Java* se diseño una nueva clase de cadenas de caracteres. Esta clase se llama *StringFast* y tiene la característica de ser más rápida uniendo cadenas de caracteres. El valor que se debe especificar indica el tamaño inicial con que se crea una cadena de caracteres.
- *saveAt*: como ya se comentó, indica cada cuanto se vuelca la información de la tabla de dispersión en la base de datos.
- *max_row_word*: indica el tamaño máximo que puede tener una palabra para ser ingresada en la tabla **words**.

JWNL section:

Esta sección permite establecer el siguiente parámetro:

- *JWNLInitFile.*

Este es el archivo de especificación de propiedades de la *JWNL* (sección 4.6.1: Tecnologías utilizadas en el servidor).

El archivo de configuración está dividido en secciones por medio de comentarios. Dicha división existe solo para aumentar la claridad del mismo y no es necesaria para el correcto funcionamiento del programa.

A continuación se muestra un ejemplo de archivo de configuración:

```
#Source section
baseFileName=c:/cats.txt

#StopWords section
stopWordsFile=C:/stopwords.txt

#Database section
dbUser=root
dbPassword=root
dbHost=localhost
dbPort=3306
database_name=odnew

#Import Parameters section
maxWords=80000
initialSize=100
saveAt=20000
max_row_word=35

#JWNL section
JWNLInitFile=C:/file_properties.xml
```

Figura 0.2 Archivo de configuración del importador de ontologías

APÉNDICE E

E

Archivo de preferencias

La aplicación cliente fue requiere guardar ciertos datos en el sistema. Para esto se genera un archivo de preferencias donde se hace persistente esta información.

El archivo consta de 5 secciones:

- Servidor
- Motores de búsqueda
- Proxy
- Idioma
- Perfiles de usuario

Servidor:

Esta sección está destinada a mantener información acerca de los servidores que dispone la aplicación. Los parámetros son los siguientes:

- *nombres*: se definen los servidores que posee el cliente.
- *actual*: establece el servidor que se está utilizando en el momento.
- Para cada servidor:
    - *url*: señala la dirección (de *Internet* o local) donde se puede acceder a las prestaciones que brinda el servidor.

Motores de búsqueda:

La sección guarda los datos relacionados con los motores de búsqueda que la aplicación cliente puede utilizar.

Se establecen los siguientes parámetros:

- *nombres*: se definen los motores de búsqueda que posee el cliente.
- Para cada nombre:
    - *query*: define la dirección de *Internet* donde va a ser envíada la consulta.

Proxies:

Aquí se especifican los datos acerca del *Proxy* que el sistema debe utilizar en caso de ser necesario.

Los parámetros a establecer son los siguientes:

- *host*: determina la dirección del *Proxy*.
- *puerto*: define el puerto al cual debe conectarse la aplicación.
- *usar*: especifica si el *Proxy* debe ser utilizado.

Idioma:

En esta sección se guarda la información relacionada con el idioma con el cual se está utilizando la aplicación.

Consta únicamente del siguiente parámetro:

- *idioma* define el idioma en que la aplicación fue utilizada por última vez, y por lo tanto, el idioma en que esta debe ser iniciada la próxima vez que se utilice.

Perfiles de usuario:

Esta sección es la más importante y hacer perdurar los perfiles que el o los usuarios locales de la aplicación hayan definido.

Los parámetros que contiene son los siguientes:

- *nombres*: guarda los nombres de cada perfil definido.
- Para cada nombre:
    - Se guarda una cadena que contiene todas las categorías marcadas por el usuario.

A continuación se especifica el formato de almacenamiento del archivo:

```
#Archivo de preferencias
#Thu Apr 21 15:27:24 GMT-03:00 2005

#Servidores
servidores.nombres=ServidorA|ServidorB (se enumeran todos los servidores definidos)
servidores.actual=ServidorA (servidor actualmente en uso, debe estar definido en la
lista de arriba)
servidores.ServidorA.url=http\://localhost\:8080/SearchServer/services/Finder
(se especifica la url de un servidor. Debe estar en la lista de nombres de
servidores)
servidores.ServidorB.url=http\://localhost

#Motores de busqueda
motores.nombres=google|yahoo|altavista
(se enumeran todos los motores de búsqueda manejados por el cliente)
motores.altavista.query=http\://www.altavista.com/web/results?q\=
(para cada motor se especifica la url para realizar consultas)
motores.google.query=http\://www.google.com/search?query\=
motores.yahoo.query=http\://search.yahoo.com/search?p\=

#Proxies
proxy.host=httpproxy (nombre del servidor proxy configurado)
proxy.puerto=3128 (puerto del servidor proxy configurado)
proxy.usar=no (indica si se está utilizando un servidor proxy o si el cliente tiene
salida directa a internet)

#Idioma
idioma=es (especifica el idioma de la interfaz gráfica)

#Perfiles de usuario
perfiles.nombres=marcos|elisa (se enumeran todos los perfiles definidos)
perfiles.marcos=Top/World|Top/Society|Top/Business
(se especifican las categorías que pertenecen al perfil)
perfiles.elisa=Top|Top/Business/Resources|Top/Business|Top/Business/Opportunities|
Top/Business/Employment
```

Figura 0.1 Archivo de preferencias

APÉNDICE F

F

Pruebas de usuarios

La experimentación del sistema se llevó a cabo a partir de un conjunto de 21 solicitudes de información. Estas solicitudes son inicialmente descritas en lenguaje natural como se puede observar a continuación.

Nº	Descripción
1	Código fuente de un foro *web* de discusión implementado en *Perl*
2	Métodos para curar el hipo
3	Alimentos para diabéticos
4	Características de la revolución francesa
5	Signos del horóscopo chino
6	Autos *Ford* de colección
7	Cuadro uruguayo que fue a Méjico 1986
8	Precios de computadoras portátiles *Pentium* 4
9	Artículos sobre ética en *Internet*
10	Forma de conectar un programa *Java* con una base de datos *MySQL*
11	Museos de *EEUU* que exponen cuadros de Torres García
12	Clase de productos que manufactura la empresa *Philips*
13	Actores principales de la película: "*A Orange Clockwork*"
14	Animales autóctonos de Australia
15	Películas en las que actuó: "*George Clooney*"
16	Empresa más grande de construcción de barcos en Japón
17	Nombre del director/decano del *MIT*
18	Cantidad de calorías de una *Big Mac*
19	Nombre del presidente de Ecuador en 1970
20	Edificio más alto de *EEUU*
21	Cantidad de dialectos hablados en la India

Figura 0.1 Descripción de las consultas

Luego son traducidas por el usuario a una consulta. A partir de estas se puede obtener el nivel de aptitud para formar consultas y clasificar a los usuarios dentro de uno de los tres conjuntos definidos, estos son: inexperto, de nivel medio y experto. Esta clasificación se realiza tomando como base la utilización de operadores avanzados en la generación de consulta y observando además el largo (la cantidad de palabras) de estas. Por una descripción más concreta de los niveles de usuario definidos y sus características, ver sección 4.13 (Experimentación).

La metodología aplicada consistió específicamente en:

- Solicitar a diferentes usuarios que generen la consulta relacionada con cada necesidad.
- Enviar la consulta al buscador "*Google*".
- Observar cuántos resultados relacionados se encontraron entre los primeros 30.
- Contabilizar cuántos resultados fueron retornados en total.
- Invitarles a que generen ahora la consulta utilizando la aplicación desarrollada.

- Enviar la consulta generada por el programa a "*Google*".
- Verificar cuántos resultados relacionados se encontraron entre los primeros 30.
- Registrar cuántos resultados fueron retornados en total.
- Comparar resultados.
- Obtener tiempo de procesamiento del servidor.

Se le dio a 5 usuarios diferentes el conjunto de solicitudes y un formulario con la siguiente información a completar:

- Consulta generada manualmente.
- Cantidad de resultados ofrecidos por *Google*.
- Cantidad de resultados relevantes entre los primeros 30.
- Consulta generada con el programa.
- Cantidad de resultados ofrecidos por *Google*.
- Cantidad de resultados relevantes entre los primeros 30.
- Tiempo de demora del servidor (tiempo que el servidor estuvo procesando información).

A partir de los formularios obtenidos se clasificó a los usuarios en los niveles antes descritos obteniendo la siguiente información para cada clase. Todas las pruebas fueron realizadas de a un usuario por vez. Esto implica que las demoras especificadas pueden cambiar si el sistema es utilizado por varios usuarios al mismo tiempo.

6.4 Usuarios expertos

Solamente un usuario fue calificado como experto y estos son los datos que obtuvo.

Nº	Consulta Consulta generada por la aplicación	Cantidad de resultados retornados	Cantidad de resultados relevantes	Demora del Servidor
1	perl forum source	1.780.000	12	3
	perl forum source (open OR expose) (languages OR language)	708.000	14	
2	hiccup cure methods	5.300	27	2
	(hiccup OR hiccough) cure (methods OR methodology OR solution)	24.700	29	
3	diabetics food	743.000	25	1
	diabetics (food OR nutrient)	794.000	25	
4	french revolution characteristics	699.000	20	2
	french revolution characteristics	699.000	20	
5	chinese horoscope signs	244.000	29	1
	(chinese OR china) horoscope (signs OR zodiac) (astrology OR astrology)	251.000	30	
6	ford classic cars	1.170.000	27	2
	ford classic (cars OR automobile OR vehicle) (antique OR antique)	1.910.000	28	
7	uruguayan team mexico 1986	14.200	2	2
	uruguayan (team OR squad OR football) mexico 1986 (soccer OR association)	14.100	1	
8	notebook pentium 4	1.960.000	28	2
	(notebook OR portable) pentium (4 OR iv) (laptops OR computer)	2.060.000	29	
9	internet ethics	14.100.000	27	2
	(internet OR cyberspace) ethics	14.400.000	27	
10	java mysql connection	1.020.000	17	1

	java mysql connection	1.020.000	17	
11	usa museums torres garcia	5.170	7	3
	usa museums torres garcia	5.170	7	
12	philips products	4.690.000	27	1
	philips products	4.690.000	27	
12	orange clockwork main actors	32.600	15	3
	orange clockwork main (actors OR performer OR artist OR actress)	82.700	12	
13	australian native animals	1.120.000	27	2
	(australian OR australia) native (animals OR fauna)	1.650.000	27	
15	george clooney movies	805.000	22	3
	george clooney (movies OR film)	770.000	19	
16	japanese ship company biggest	914.000	3	2
	japanese ship company (biggest OR greatest OR largest) management	575.000	10	
17	mit university president	3.750.000	12	2
	(mit OR massachusetts) university (president OR director) institutes (research OR investigation)	678.000	4	
18	big mac fats	87.200	24	2
	big mac fats	87.200	24	
19	ecuadorian president 1970	10.300	9	3
	ecuadorian president 1970	10.300	9	
20	usa biggest building	2.800.000	1	2
	(usa OR us) (biggest OR greatest OR largest) (building OR construction) (model OR mock)	19.200.000	2	
21	india dialects	233.000	24	1
	india (dialects OR idiom OR speech OR language)	20.500.000	26	

Figura 0.2 Usuario nº1

6.5 Usuarios de nivel medio

Los resultados de las evaluaciones para usuarios de nivel medio son las siguientes.

Nº	Consulta Consulta generada por la aplicación	Cantidad de resultados retornados	Cantidad de resultados relevantes	Demora del Servidor
1	perl source code forum	930.000	7	2
	perl source (code OR microcode) forum	928.000	9	
2	cure hiccups	48.500	29	1
	cure (hiccups OR hiccough)	67.900	29	
3	diabetics food	743.000	25	1
	diabetics (food OR nutrient)	794.000	25	
4	"french revolution"	1.270.000	28	1
	french revolution	8.820.000	27	
5	"chinese horoscope"	136.000	28	1
	(chinese OR china) horoscope	671.000	30	
6	ford collector	722.000	25	1

	ford collector (antique OR antique) (classic OR artist)	206.000	28	
7	uruguay soccer futbol world cup 1986	738	0	2
	uruguay (soccer OR football) futbol world cup 1986 (players OR participant)	618	0	
8	pentium iv prices	277.000	29	1
	pentium iv prices comparisons	10.200	21	
9	ethics papers articles	2.290.000	1	1
	ethics (papers OR document) (articles OR writing) (internet OR net)	3.270.000	9	
10	java mysql connectivity	444.000	12	1
	java mysql connectivity (databases OR database)	381.000	16	
11	museum torres garcia	37.700	12	2
	museum torres garcia	37.700	12	
12	philips products	4.690.000	27	1
	philips products	4.690.000	27	
13	orange clockwork	634.000	4	1
	orange clockwork	634.000	4	
14	australia indigenous fauna	164.000	6	1
	australia (indigenous OR autochthonal OR autochthonic OR autochthonous OR native) (fauna OR animal)	1.740.000	7	
15	george clooney	829.000	21	1
	george clooney (movies OR movie) (acting OR acting)	102.000	13	
16	japan ship manufacturing	676.000	1	1
	(japan OR nippon OR japanese) (ship OR transport) (manufacturing OR fabricate OR make OR create) (boats OR vessel)	986.000	1	
17	mit dean	733.000	0	0
	(mit OR massachusetts) (dean OR head OR chief) (students OR pupil)	3.999.000	2	
18	big mac	17.600.000	0	0
	big mac	17.600.600	0	
19	ecuador president 1970	122.000	17	2
	(ecuador OR ecuadorian OR ecuadoran) (president OR chief) 1970	155.000	17	
20	tallest building in the united states	217.000	16	1
	tallest building united states	217.000	16	
21	india language dialects	182.000	17	1
	india language (dialects OR idiom) (speech OR communication)	145.000	20	

Figura 0.3 Usuario nº2

Nº	Consulta Consulta generada por la aplicación	Cantidad de resultados retornados	Cantidad de resultados relevantes	Demora del Servidor
1				
2	hiccup cure	29.200	25	0
	hiccup cure	29.200	25	
3	diabetics food	664.000	24	1
	diabetics (food OR nutrient OR substance)	683.000	21	
4	french revolution	8.820.000	27	0

	french revolution (social OR societal) (movements OR front)	2.170.000	23	
5	chinese horoscope	841.000	29	0
	(chinese OR china) horoscope	671.000	30	
6	ford cars collection	865.000	10	1
	ford (cars OR auto OR automobile OR motorcar OR vehicle OR automotive) collection	1.840.000	6	
7	uruguay team football soccer méxico 1986	7.230	1	3
	uruguay (team OR squad) football soccer méxico 1986	7.690	2	
8	prices laptop pentium 4	2.820.000	26	2
	prices (laptop OR portable) pentium (4 OR iv)	3.320.000	27	
9	ethics articles	12.000.000	0	1
	ethics articles (internet OR net)	5.070.000	12	
10	how to connect a java program with mysql database	268.000	9	2
	how connect java (program OR programme) mysql database	274.000	15	
11	torres garcía usa museums	6.920	5	2
	torres garcía usa museums (north OR geographical) america	4.730	5	
12	philips products	4.690.000	27	0
	philips products	4.690.000	27	
13	orange clockwork movie	684.000	5	1
	orange clockwork (movie OR film)	842.000	10	
14	australian animals	3.760.000	22	2
	australian (animals OR creature OR fauna) australia (environment OR environment)	937.000	26	
15	george clooney movies	805.000	22	1
	george clooney (movies OR film)	770.000	19	
16	japan biggest shipbuilding company	26.100	7	3
	japan (biggest OR greatest OR largest) shipbuilding (company OR establishment) (naval OR navy) (engineering OR technology)	27.300	8	
17	mit dean	733.000	0	1
	(mit OR university) (dean OR head OR leader)	150.000.000	0	
18	big mac calories	152.000	23	1
	big mac calories	152.000	23	
19	equator president 1970	29.000	0	1
	equator president 1970	29.000	0	
20	highest building usa	5.130.000	5	1
	highest building usa	5.130.000	5	
21	indians dialects	89.200	2	0
	indians (dialects OR idiom OR speech OR spoken OR language)	4.100.000	0	

Figura 0.4 Usuario nº3

6.6 Usuarios inexpertos

Estos son los resultados para usuarios inexpertos. Ellos consideraron que la demora del servidor era despreciable y por lo tanto no rellenaron tal columna.

Nº	Consulta Consulta generada por la aplicación	Cantidad de resultados retornados	Cantidad de resultados relevantes	Demora del Servidor
1	forum perl source code	1040000	5	
	forum perl source code (languages OR language) software	727000	7	
2	hiccup	930000	4	
	hiccup health (conditions OR condition) (diseases OR disease)	21400	8	
3	diabetic food	1710000	21	
	diabetic food health (cooking OR cookery)	909000	21	
4	french revolution characteristics	628000	14	
	french revolution characteristics (history OR past) aspect	259000	17	
5	chinese horoscope sign	436000	20	
	chinese horoscope (sign OR zodiac) (astrology OR star) (divination OR foretelling)	41300	14	
6	ford old car	4060000	15	
	ford old (car OR auto OR automobile OR motorcar) (custom OR customs) (collector OR electrode) (trucks OR truck) vehicle	96300	19	
7	uruguayan team mexico 1986	2660	4	
	uruguayan (team OR squad) mexico soccer 1986 national football game	305	6	
8	notebook pentium 4 price	947000	15	
	notebook OR computer) pentium (4 OR iv) price (laptops OR laptop) portable pc	757000	12	
9	internet ethics	14000000	14	
	internet (ethics OR moral OR philosophy)	37100000	10	
10	Java mysql connect	558000	12	
	java mysql (connect OR link) (programming OR programing) (software OR system) language connection	160.000	8	
11	torres garcia museum usa	61000	4	
	torres garcia museum (usa OR united OR states OR america OR u.s. OR u.s.a.) (arts OR humanistic)	23500	7	
12	philips kind products	617000	8	
	philips (kind OR sort OR variety) products	3700000	11	
13	actors orange clockwork	134000	18	
	actors orange clockwise (actresses OR actress) (movies OR movie) (performing OR acting) (arts OR humanistic)	412	4	
14	native australian animals	1360000	28	
	native australian animals (zoology OR zoological) (oceania OR oceanica) (biodiversity OR diverseness) australia	571	12	
15	george cloney movies	736000	26	
	george clooney (movies OR film OR picture) (titles OR title) clooney,	285000	28	
16	japan biggest ship building company	449000	2	
	(japan OR nippon) (biggest OR greatest OR largest) ship building company (maritime OR nautical) asian	63200	4	
17	mit director	11200000	2	
	mit director	11200000	2	
18	big mac calories	215000	14	
	big mac calories	215000	14	
19	ecuador president 1970	135000	7	

	ecuador president 1970 (south OR geographical) america	108000	8	
20	tallest building usa	197000	16	
	tallest building usa (united OR unify) (states OR state) high	87000	8	
21	india dialects	227000	6	
	india (dialects OR idiom OR accent) (linguistics OR science) regions	99100	8	

Figura 0.5 Usuario nº4

Nº	Consulta Consulta generada por la aplicación	Cantidad de resultados retornados	Cantidad de resultados relevantes	Demora del Servidor
1	Forum source web perl	1.440.000	20	
	perl forum source web (open OR expose) (languages OR language)	802,000	23	
2	cure hipo method	978	4	
	(hiccup OR hiccough) (cure OR heal) method (conditions OR condition) (diseases OR disease) solution	6,150	28	
3	food for diabetics	740,000	21	
	food diabetics (nutrition OR organic) person diseased	6,080	29	
4	Characteristics of the French revolution	3,360,000	10	
	characteristics french revolution (social OR society) (history OR account)	663,000	30	
5	Signs of horoscope Chinese	758,000	22	
	(signs OR zodiac) horoscope chinese astrology divination	77,200	30	
6	Ford cars of collection	844,000	10	
	ford (cars OR auto OR automobile OR machine OR motorcar) collection (arts OR humanistic) (art OR fine) automotive	136,000	30	
7	Uruguayan team that went to Mexico 1986	821	16	
	uruguayan team that went mexico 1986 (soccer OR association) (players OR player) (national OR subject) uruguay	382	28	
8	Prices of portable computers Pentium 4	1,700,000	21	
	Prices of portable computers Pentium 4	993,000	26	
9	Articles on ethics in Internet	6,910,000	17	
	articles ethics internet	6,920,000	20	
10	Form to connect a Java program with a data base MySQL	375,000	6	
	form connect java program data base mysql (internet OR net) (computers OR computer) connection	228,000	21	
11	Museums of the U.S.A. that exposes pictures of Torres Garcia	166	0	
	museums u.s.a. that exposes (pictures OR pictorial) torres garcia (art OR artistic) (history OR humanistic)	141	1	
12	Clase de productos que manufactura la empresa Philips	526	6	
	product class that (manufactures OR fabrication) philips company creating fabricate institution	895	7	
13	Film stars of the film: "Orange Clockwork"	1,840	15	
	(film OR movie OR picture) (stars OR principal) film: "orange clockwork" (titles OR title) actor	87	21	
14	Native animals of Australia		30	

	native animals australia (zoology OR zoological)	138,000	30	
15	Films in which act "George Clooney"	73,900	2	
	films which act "george clooney" movie work play	31,200	12	
16	Greater company of construction of boats in Japan	357,000	3	
	greater company (construction OR building) boats japan	156,000	16	
17	name of the director/decano of the MIT	0	0	
	name director (mit OR massachusetts OR institute OR technology) (science OR scientific)	25,000,000	24	
18	Amount of calories of a Big Mac	68,300	29	
	amount calories big mac organizations	39,100	30	
19	Name of the president of Ecuador in 1970	87,300	15	
	name president (ecuador OR republic) 1970 (history OR account) country	516,000	24	
20	Higher building of the U.S.A.	38,100,000	0	
	higher (building OR construction) u.s.a. (technology OR engineering) maintenance high commercial	1,920,000	0	
21	Amount of dialect spoken in India	43,900	21	
	amount (dialect OR idiom) spoken india society	46,900	30	

Figura 0.6 Usuario nº5

6.7 Comparación y análisis

A continuación se presenta una gráfica que muestra las variaciones en las cantidades de resultados obtenidas al utilizar la aplicación.

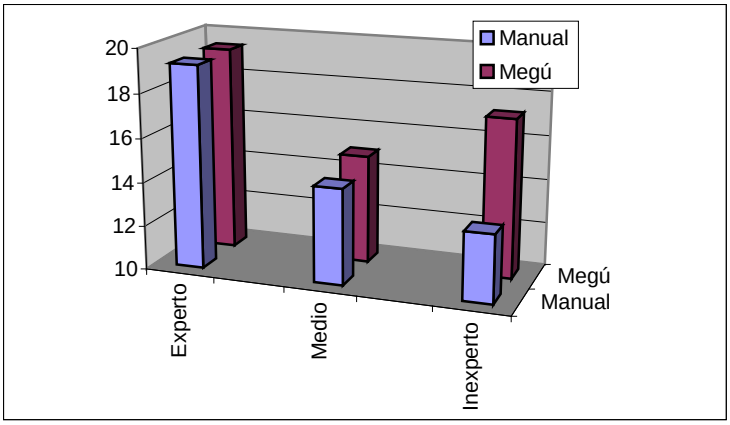

Figura 0.7 Resultados obtenidos

La figura muestra dos columnas para cada nivel de usuario. La primera de ellas muestra la cantidad de resultados relevantes encontrados generando la consulta manualmente, y la segunda el número de resultados relevantes obtenidos generando la consulta con la aplicación.

Estos resultados parecen alentadores, ya que el efecto de utilizar la aplicación mejora la cantidad de documentos relevantes obtenidos. Sin embargo, es muy extraño que un usuario inexperto obtenga una mayor cantidad de resultados relevantes utilizando la aplicación que uno de nivel medio. Esto aparentemente sucede por las siguientes causas:

1. Los usuarios no son capaces de discriminar correctamente cuales documentos son relevantes.

2. La cantidad de datos no es lo suficientemente significativa.
3. La evaluación es muy extensa y hace que esta no sea tomada con la suficiente seriedad.

Entonces, la metodología de evaluación no arroja los resultados esperados para el esfuerzo que se requiere en completar el formulario.

Dada la subjetividad de lo que se desea cuantificar se requiere de un análisis estadístico en gran escala para obtener datos significativos. Esta tarea escapa al alcance de este trabajo.