Integración de Conocimiento Web en Sistemas de Razonamiento Basado en Casos

PROYECTO DE FIN DE MÁSTER EN SISTEMAS INTELIGENTES

Miguel Ángel Casado Hernández
Directora: Belén Díaz Agudo
Co-director: Juan Antonio Recio García

Departamento de Ingeniería del Software e Inteligencia Artificial
Máster en Investigación en Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2009-2010

Integración de Conocimiento Web en Sistemas de Razonamiento Basado en Casos

Memoria de Proyecto de Fin de Máster

Departamento de Ingeniería del Software e Inteligencia Artificial
Máster en Investigación en Informática
Facultad de Informática
Universidad Complutense de Madrid

Curso 2009-2010

Resumen

El Razonamiento Basado en Casos o CBR (del inglés Case-Based Reasoning) se basa en la utilización de experiencias previas para resolver nuevos problemas. Los últimos congresos referentes a esta materia han señalado la Web como una enorme fuente de información donde usuarios dejan sus experiencias en blogs, wikis, etc. Nuestro trabajo comienza con un estudio a nivel teórico de los aspectos estructurales y funcionales que se deberían modificar para la inclusión de este tipo de información en un sistema CBR tradicional. Posteriormente, realizaremos un estudio en busca de una fuente de información online para poder explotar y ser capaces de integrar en el framework jCOLIBRI2. Expondremos y ejemplificaremos una serie de algoritmos para extraer datos de la fuente online elegida, tanto para ser utilizados en solitario como para emplearlos en combinación con alguna aplicación que trabaje únicamente con fuentes de conocimiento locales. Para concluir se realizarán una serie de experimentos con el objetivo de comprobar la mejora de una base de casos local con casos obtenidos de la Red.

Palabras clave:

- Web-CBR
- jCOLIBRI2
- Freebase
- Base de casos Web
- Contenedores de conocimiento

Abstract

Case-Based Reasoning (CBR) is based in the usage of previous experiences in order to resolve new problems. Recent conferences regarding this subject have pointed to the Web as an interesting source of information where users share their experiences in blogs, wikis, etc. Our work begins with a theoretical study about structural and fuctional aspects that should be modified in order to include this kind of information. Subsequently we will perform an study to search an online source of information to explote and be able to integrate into the jCOLIBRI2 framework. We will present and illustrate a set of algorithms to extract data from the chosen online source, to be used alone or to be utilized in combination with an application that works only with local data. To conclude, a set of experiments will be done with the objective of check if a local case base can be improved with cases extracted from the Web.

Keywords:

- Web-CBR
- jCOLIBRI2
- Freebase
- Web Case Base
- Knowledge containers

Índice

Resumen v

Abstract vii

1. Introducción **1**
1.1. Motivación y objetivos 1
1.2. Estructura del documento 3

2. Marco Teórico **5**
2.1. Introducción al CBR 5
2.1.1. El ciclo CBR 6
2.1.2. Knowledge containers 7
2.1.3. Estructura de la base de casos 10
2.1.4. WEB-CBR 11
2.1.5. Aproximaciones previas al WEB-CBR 12
2.1.6. CBR Multiagente 13
2.2. Modelo teórico para WEB-CBR 15
2.2.1. Aspectos estructurales 15
2.2.2. Aspectos funcionales 18
2.2.3. Gestión y mantenimiento de la base de casos 20
2.3. Conclusiones 21

3. Fuentes Web **23**
3.1. Introducción 23
3.2. Estudio de diferentes fuentes Web 24
3.2.1. Google Squared 24
3.2.2. Yahoo! Query Language 25
3.2.3. SearchMonkey 25
3.2.4. Freebase 26
3.3. Discusión: ventajas y desventajas 26
3.3.1. La Organización de Freebase 27
3.4. Conclusiones 29

4. Marco Práctico **31**
4.1. Introducción . . . 31
4.2. Dominio de trabajo . . . 32
4.2.1. Computer Cooking Contest . . . 32
4.2.2. Food & Drink en Freebase . . . 33
4.3. Algoritmos . . . 34
4.3.1. WebRetrieval . . . 35
4.3.2. CombinedRetrieval . . . 37
4.3.3. WebAdaptation . . . 38
4.4. Integración en jCOLIBRI2 . . . 40
4.4.1. Introducción a jCOLIBRI . . . 40
4.4.2. Integración . . . 42
4.5. Conclusiones . . . 46

5. Validación Experimental **47**
5.1. Hipótesis . . . 47
5.2. Primer experimento . . . 47
5.3. Segundo Experimento . . . 51
5.4. Conclusiones . . . 53

6. Conclusiones y trabajo futuro **55**
6.1. Conclusiones . . . 55
6.2. Trabajo futuro . . . 57

A. Normalización de tipos de cocina **59**

Bibliografía **61**

Índice de figuras

2.1. El Ciclo CBR 7
2.2. Espacio de problemas y de soluciones. 8
2.3. Distribuciones de sistemas CBR según número de agentes y número de bases de casos. Figura obtenida de Plaza y McGinty (2005). 14
2.4. Ciclo CBR para base de casos web única. 17
2.5. Ciclo CBR para modelo mixto de base de casos. 18

4.1. Ejemplo de plato de Freebase. 34
4.2. Integración del conector para Freebase. 43

5.1. Resultados para Leave One Out sobre base local. 49
5.2. Resultados para Leave One Out sobre base web. 49
5.3. Resultados para Leave One Out sobre base mixta. 50
5.4. Modelo a seguir en el segundo experimento. 51
5.5. Evaluación de queries de ColibriCook sobre Freebase. Cocinas normalizadas. Mismo número de recetas. 52
5.6. Evaluación de queries de Freebase sobre ColibriCook. Cocinas normalizadas. Mismo número de recetas. 52
5.7. Resultados del segundo experimento. 53

Capítulo 1

Introducción

Resumen: En este Capítulo se presenta una breve introducción al proyecto. Comenzaremos por la motivación que impulsó la realización del trabajo y expondremos también una estructura general del documento, indicando en qué consiste cada uno de sus Capítulos.

1.1. Motivación y objetivos

El Razonamiento Basado en Casos (CBR, del inglés Case-Based Reasoning) propone la reutilización de las experiencias pasadas -casos- para resolver nuevos problemas. A grandes rasgos, esta aproximación se basa en la recuperación y adaptación de las soluciones de dichas experiencias anteriores para que puedan ser aplicadas a la situación actual [Aamodt y Plaza (1994), de Mantaras et al. (2006)]. Este tipo de técnicas se fundamenta en la suposición de que problemas similares tendrán soluciones similares y que los problemas tienden a recurrir. Con esto, evitamos tener que construir soluciones desde cero. El elemento fundamental, por tanto, es el mantenimiento de una buena base de casos que contenga suficientes experiencias previas para permitir solucionar los nuevos problemas.

Uno de los problemas del CBR clásico radica en la obtención de casos. Se precisa de la búsqueda de experiencias y su estructuración en forma de casos hasta construir una buena base de casos. Las últimas conferencias sobre CBR han propuesto la Web como una gran fuente de conocimiento para poder explotar en este tipo de aplicaciones [Plaza (2008)]. En la red, los usuarios publican sus experiencias y opiniones sobre toda clase de ámbitos. Toda esta información puede ser considerada como un diamante en bruto para el Razonamiento Basado en Casos. Su explotación supondría un gran avance en la materia. Hasta el momento, la obtención de información de blogs, wikis, etc. se realizaba mediante técnicas de Extracción de Información textual

[Asiimwe et al. (2007), Wiratunga et al. (2004), Wiratunga et al. (2006)]. Al optar por una representación estructurada de los casos, la recuperación de información estructurada sobre fuentes de datos no estructuradas es muy costosa. Debido a eso, grandes compañías como Google o Yahoo! comienzan a proporcionar herramientas de adquisición de información de manera ya estructurada desde la misma Web. La integración de estas herramientas en los sistemas CBR evitaría el inconveniente de la estructuración de información para crear la base de casos, encontrado hasta ahora en esta disciplina.

Nuestro trabajo comenzará con el estudio de estas nuevas fuentes de información hasta decantarnos por una para intentar integrarla en el framework CBR JCOLIBRI2,desarrollado por el grupo de investigación GAIA[1] de la Universidad Complutense de Madrid y que es actualmente una de los más extendidos para el desarrollo de sistemas CBR. De este modo, esta nueva funcionalidad estaría disponible para la creación de nuevas aplicaciones CBR. Desarrollaremos los aspectos teóricos y prácticos que conllevaría esta posible integración y finalizaremos con un estudio comparativo con aplicaciones CBR clásicas.

A continuación, presentamos los objetivos del proyecto de manera esquematizada:

- Realizar un estudio de las difrentes fuentes web que ofrecen conocimiento de manera estructurada, prestando especial atención a la viabilidad de una posible integración en el framework de JCOLIBRI2.

- Exponer de manera teórica los cambios que supondría la utilización de una fuente externa tan grande como la Web en los Sistemas Basados en Casos. Asimismo, explicar los beneficios e inconvenientes que supondría el uso de dicha fuente de información.

- Construcción de un conector para la fuente web seleccionada e integración en JCOLIBRI, dejando disponible esta nueva funcionalidad a futuros diseños de sistemas CBR.

- Proposición de una serie de algoritmos teóricos para la comprensión y utilización de la nueva funcionalidad diseñada.

- Creación de una aplicación CBR sobre un dominio seleccionado. Estudio de ventajas y desventajas de este nuevo sistema realizando comparaciones con los sistemas clásicos CBR que emplean bases de casos locales.

[1] http://gaia.fdi.ucm.es/

1.2. Estructura del documento

La memoria de nuestro proyecto está organizada en las siguientes Secciones:

- En el siguiente Capítulo explicaremos el marco teórico en el que se estructura nuestro proyecto.
- En el tercer Capítulo expondremos un estudio de diversas fuentes web para la obtención de conocimiento. Explcaremos cómo sería posible acceder a ellas y las posibilidades de integrarlas en jCOLIBRI2.
- En el cuarto Capítulo comenzaremos proponiendo una hipótesis y los algoritmos necesarios para poder resolverla. Dichos algoritmos obtendrán información de la fuente web elegida y serán ejemplificados para su mejor comprensión.
- En el Capítulo quinto detallaremos las modificaciones necesarias para la integración de Freebase en jCOLIBRI así como la implementación del conector necesario para hacerlo funcionar.
- El sexto Capítulo propondremos una serie de experimentos, para intentar dar respuesta a la hipótesis planteada en el Capítulo 4, detallando sus resultados y explicando las conclusiones alcanzadas.
- Para finalizar redactaremos las conclusiones alcanzadas tras terminar este proyecto y enumeraremos los distintos puntos sobre los que proseguir en un futuro trabajo.

Se ha omitido la sección de *Estado del arte*, porque se ha ido desglosando en cada Capítulo. Entendemos que de este modo queda más claro, explicando al comienzo de cada sección los trabajos relacionados necesarios para la comprensión del resto del Capítulo.

Capítulo 2

Marco Teórico

Resumen: En este Capítulo expondremos una breve introducción al CBR, indicando los aspectos más importantes tratados en nuestro proyecto. A continuación explicaremos los cambios a nivel teórico que propondremos para añadir información de una fuente de datos online al Razonamiento Basado en Casos.

2.1. Introducción al CBR

El Razonamiento Basado en Casos (CBR) es un paradigma de resolución de problemas y aprendizaje que difiere de la mayoría de aproximaciones de Inteligencia Artificial. Es capaz de resolver problemas sin necesidad de poseer un conocimiento general del dominio de trabajo. Se basa de experiencias previas, denominadas *casos*, que recuperará mediante diferentes medidas de similitud y adaptará hasta poder resolver el problema actual. En vez de emplear reglas, utiliza conocimiento específico de situaciones pasadas, para intentar obtener la solución buscada. Denominamos caso a una experiencia previa que ha sido capturada y aprendida para poder ser reutilizada con el fin de resolver un futuro problema. Cada problema resuelto es almacenado para una futura utilización, logrando así *aprendizaje* en estos sistemas [Aamodt y Plaza (1994)]. También es posible lograr aprendizaje fijándose en qué soluciones han fallado a la hora de resolver un problema y así evitarlas en un futuro.

Nacidos a finales de los 70 (aunque la primera aplicación CBR, *CYRUS* [Kolodner (1983)], data de 1983), se proponen como alternativa a los sistemas basados en reglas. Este tipo de sistemas evita la necesidad de una generalización y obtención de reglas a partir de informaciones concretas, que también es aprendizaje. Los CBR se fundamentan en una suposición: *problemas similares tendrán soluciones parecidas*. Por lo tanto, si conseguimos obtener un

problema resuelto semejante a la consulta actual, podremos adaptarlo hasta obtener una nueva solución. Con esto, queremos destacar la importancia de disponer de una buena colección de experiencias previas o casos para poder reutilizar y adaptar. Decimos que los casos se almacenan en una *base de casos*, que será un término muy empleado a lo largo de este trabajo.

Este tipo de tecnología está muy extendida, pudiendo encontrarla en aplicaciones como recomendadores, p.ej. *Analog Devices Inc.*; herramientas de soporte técnico, como *3Com KnowledgeBase*; diagnóstico de enfermedades, p.ej. *Protos*; sistemas de predicción, véase *Shai*, ...

2.1.1. El ciclo CBR

Una vez que hemos comentado en qué consiste el Razonamiento Basado en Casos, vamos a explicar de manera más detallada el proceso por el que pasa una consulta de un usuario hasta ser resuelta. Las aplicaciones CBR se guían por un proceso cíclico introducido por Aamodt y Plaza (1994): Tras la llegada de una consulta, se buscan los casos resueltos más similares a éste. Se intenta adaptar la solución del problema previo al problema actual. Finalmente, se revisa y se almacena la nueva solución. Por tanto, el proceso puede describirse, a nivel general, como un ciclo de cuatro etapas bien diferenciadas, comunmente llamadas las 4 R's (Figura 2.1).

1. RECUPERAR: Mediante funciones de similitud entre casos, se recuperan de la base de casos las experiencias previas más similares al problema que acaba de llegar. Los casos recuperados suelen estar ordenados en función de las medidas utilizadas por las anteriores funciones de similitud.

2. REUTILIZAR: Se identifican las diferencias entre la consulta y los casos recuperados y gracias a funciones de adaptación, se ajusta la solución del problema recuperado de la base de casos al problema que estamos tratando ahora. Si el problema actual ya se encontrara en la base no se precisaría de adaptación.

3. REVISAR: Esta etapa consiste en recibir el visto bueno de la nueva solución adaptada. Esta etapa suele corresponder al usuario externo, que decide si la respuesta es de su agrado o no.

4. RECORDAR: Por último, dado que los sistemas CBR son sistemas de aprendizaje, guardaremos el problema y la solución en nuestra base de casos, para poder emplearlas en un futuro.

En la parte central de la imagen del ciclo CBR, se puede observar que la base de casos está acompañada de lo que se ha denominado *conocimiento general*. Aquí se ubica el conocimiento necesario para completar la ejecución

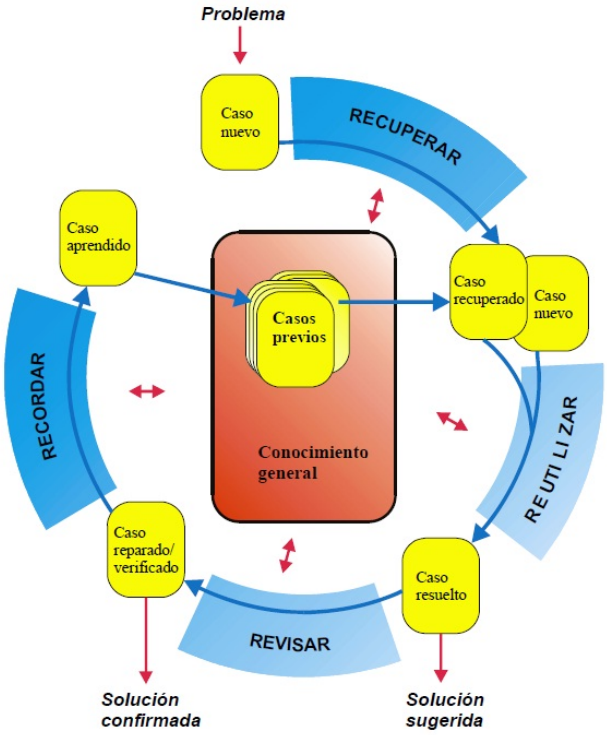

Figura 2.1: El Ciclo CBR

del proceso y que no compete al almacenamiento de casos, p.ej. las funciones de similitud y adaptación que mencionamos anteriormente.

2.1.2. Knowledge containers

Como acabamos de contar, existe más conocimiento que el comprende la base de casos y que hemos denominado conocimiento general. Existe una clasificación propuesta en Richter (1995) que distingue el conocimiento empleado por una aplicación CBR agrupándolo en los denominados *knowledge containers* (*kc*). Estos son los *kc* que se utilizan más frecuentemente:

- Los casos almacenados, que denominaremos kc_{cb}.
- Las funciones de similitud empleadas para la recuperación de casos en función de la consulta o *query*, que llamaremos kc_{ret} y que se denotan como R en la Figura 2.2.
- El conocimiento de adaptación: el conjunto de reglas que se emplean para transformar la solución del caso recuperado de la base de casos

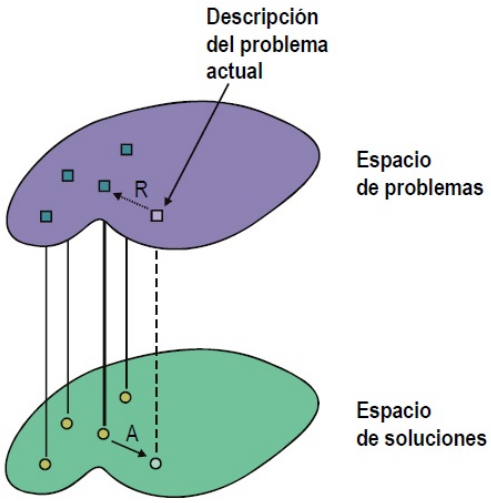

Figura 2.2: Espacio de problemas y de soluciones.

en una solución válida para el nuevo problema. Lo designaremos como kc_{adapt}. La Figura 2.2 representa el proceso de adaptación de una solución como la distancia A necesaria para su transformación.

A continuación, vamos a explicar de manera más detallada los elementos más importantes de cada uno de los *knowledge containers* anteriores.

2.1.2.1. Los casos

El *knowledge container* kc_{cb} es el que representa los casos almacenados. Hemos dicho al comienzo del Capítulo que entendíamos como caso a una experiencia previa que ha sido capturada y aprendida para poder ser reutilizada con el fin de resolver un futuro problema. Es, por consiguiente, la base sobre la que se construye una aplicación CBR. Como se explica en Díaz-Agudo (2002), las características de los casos dan pie a una división en cuatro partes:

- *Despripción*: Corresponde con la descripción del problema que se resuelve. En esta parte es dónde se ubican las consultas de los usuarios.
- *Solución*: información necesaria para alcanzar los objetivos que aparecen en la descripción.
- *Resultado*: resultado de aplicar la solución. También se guarda aquellas soluciones que fallaron para evitar caer otra vez en el mismo error.

- *Justificación*: trazas que han llevado a la solución y explicaciones de por qué se ha optado por determinada opción o no.

En Althoff et al. (1995), se explica que cada una de estas cuatro partes se representa mediante atributos, pudiéndose utilizar un modelo *plano*, a modo de lista de atributos, o un modelo *estructurado*, con estructuras más complejas para organizar estos atributos.

2.1.2.2. Funciones de Similitud

El kc_{ret} es el conocimiento utilizado a la hora de decidir cuáles son los casos que más se parecen a la consulta que llega al ciclo CBR. Principalmente, lo forman un conjunto de funciones de similitud que se encargan de la comparación entre casos proporcionando algún tipo de medida. Estas funciones deben poder comparar dos casos, independientemente de su estructura. En el apartado anterior, dijimos que la descripción de un caso estaba compuesta por atributos, que podían estar agrupados de forma lineal o en algún tipo de estructura más compleja.

Dividimos la medición de similitud en dos partes:

- La similitud local, que es la encargada de medir la similitud entre atributos.
- La similitud global, que devuelve ya la medida final de similitud entre dos casos, teniendo en cuenta la estructura de los mismos y basándose en la similitud local.

2.1.2.3. Funciones de Adaptación

El *knowledge container* que nos queda por explicar de los expuestos anteriormente, es el kc_{adapt}. Se trata del conocimiento que precisa un sistema CBR para realizar la adaptación se los casos recuperados al problema actual, realizada en la fase de *Reutilización* del ciclo CBR (Figura 2.1).

La misión de la adaptación puede verse de manera más clara en la Figura 2.2. Una vez encontrado el caso más similar a la consulta entrante, intentamos adaptar la solución de dicho caso hasta obtener una posible solución para la consulta. Destacamos que es posible que no se requiera una adaptación si, por ejemplo, la consulta se encuentra ya en la base de casos. También suele darse este caso en tareas de clasificación que no requieren una modificación de la solución.

Cualquier diferencia entre la consulta y el caso recuperado puede suponer la modificación de la solución. En de Mantaras et al. (2006) se hace una clasificación de los métodos de adaptación en tres grupos:

- *Sustitución* de algunas de las partes de la solución del caso recuperado. Esta técnica es bastante recurrente en sistemas CBR del dominio de las

recetas de cocina en los que se dan adaptaciones en forma de sustitución de ingredientes.

- *Transformación* de la solución alterando la estructura de la misma. Si volvemos al ejemplo de las recetas de cocina, una sustitución puede conllevar la eliminación de algún paso en la preparación del plato de la nueva solución. Se está alterando la estructura de la solución del caso recuperado.

- *Adaptación generativa*, que no adapta la solución directamente. Deriva la solución repitiendo el proceso que dio lugar a la solución del caso recuperado.

Para las dos primeras opciones, se precisa conocer cómo afectarán las diferencias entre consulta y caso recuperado en la adaptación de la solución. Se necesita, por tanto, conocimiento de adaptación. La adquisición de éste es un proceso costoso que varios autores han intentado lograr de manera automática mediante diversas aproximaciones como en Hanney y Keane (1997). Estos autores proponen un algoritmo basándose en la comparación de casos. Intentan inferir una serie de reglas, que se pueden combinar entre sí, para deducir qué adaptaciones son viables o no, ayudándose de una serie de funciones heurísticas. Tienen una suposición en la que se apoyan en su trabajo, y es que *los casos de la base de casos son representativos de futuros problemas objetivo*. El principal problema que admiten tener es que no saben si su método podrá ser usado con estructuras complejas a la hora de representar casos. Estas ideas han sido utilizadas en otras aplicaciones, como d'Aquin et al. (2007), que trabaja en el dominio médico del cáncer. Se pueden consultar más aproximaciones como Wilke et al. (1996) o McSherry (1999).

2.1.3. Estructura de la base de casos

Una vez explicada la estructura de los casos, es necesario decidir la organización de los casos dentro de la base de casos para facilitar su recuperación. Con el fin de acelerar el proceso de comparación entre casos y conseguir recuperar los más relevantes podemos organizar la base de casos de diferentes maneras, desde una simple lista de casos hasta estructuras más complejas como árboles k-d, redes de recuperación, redes de discriminación, ...

En estas estructuras más complejas es preciso determinar qué es lo que se va a comparar para ir guiando el proceso de recuperación, es decir, se necesitan determinar los *índices* en busca de mejorar la eficiencia. La identificación de los atributos que se emplearán como índices se puede realizar de diferentes maneras como consultando a expertos, seleccionando aquellos atributos que se quieren resolver, ... En Díaz-Agudo (2002) encontramos una serie de organizaciones de la base de casos:

- Árboles de decisión que intentan minimizar el número de pasos de la fase de recuperación.
- Árboles k-d que están diseñados para la recuperación *k-nearest neighbours*, que no recuperan sólo el caso más cercano, sino un conjunto de k más casos semejantes.
- Redes de activación, que son una adaptación de redes neuronales donde mediante interconexión de nodos se procede a la recuperación utilizando técnicas de activación y propagación de nodos.
- Otro tipo de redes de nodos en las que los nodos más internos son generalizaciones de los que se encuentran en los extremos.
- Redes de categorías, relaciones semánticas, casos e índices.
- Aproximaciones que utilizan modelos taxonómicos y terminológicos de los dominios de trabajo.

2.1.4. WEB-CBR

Como explicábamos anteriormente, el CBR se fundamenta en la reutilización de experiencias previas para la resolución de nuevos problemas. Una de las cuestiones a tener en cuenta a la hora de construir un sistema CBR es la disposición o no de una base de casos. La construcción de una base de casos puede ser un trabajo costoso, sobre todo cuando se debe hacer desde cero. No obstante, si buscamos una fuente de experiencias, de casos con descripción y solución, rápidamente encontramos una inmensa: la Web. ¿Y si a la hora de buscar casos ya resueltos pudieramos servirnos de la mayor fuente de información que hay en el mundo? Esta cuestión, ya tratada en congresos como el ICCBR[1], es uno de los aspectos más emergentes en esta materia. Se denomina WEB-CBR.

Ya hemos explicado que nuestra idea es intentar explotar la ingente cantidad de experiencias que hay en la Web. Sin embargo, todas estas experiencias están tratadas como documentos, pero son experiencias que deberían proporcionarnos conocimiento de experiencias (*Experiential Knowledge*) y debería ser representado como tal.

Antes de embarcarnos en lo que es concretamente la propuesta de nuestro trabajo, hemos de explicar brevemente el concepto de la Web Semántica, introducido por Berners-Lee et al. (2001) y revisado en Shadbolt et al. (2006). La Web, originariamente, estaba diseñada para ser leída e interpretada por humanos. Sin embargo, la idea de que una máquina pudiera recorrer sitios web para recolectar información y realizar tareas más sofisticadas desembocó en lo que se conoce como Web Semántica (SW de *Semantic Web*). Se dice que

[1] International Conference on Case-Based Reasoning

la SW es una extensión de la *World Wide Web* original en el sentido de que se comenzó a construir añadiendo *metadatos* a los sitios web, proporcionando así significados y relaciones entre datos.

Para Plaza (2008) y Plaza y Baccigalupo (2009), existen dos enfoques dentro de la Web Semántica:

- La visión *top-down* basada fundamentalmente en ontologías. Las ontologías son documentos que definen formalmente relaciones entre términos de un determinado dominio. Los agentes o máquinas que recorran la Web, pueden ayudarse de esta clasificación que alguien ha diseñado. Este concepto de que *alguien ha diseñado la ontología* es a lo que Plaza se refiere cuando define como *top-down*: alguien define y mantiene la ontología de un dominio.

- El enfoque *bottom-up*, en el que destacan las *folksonomías*. El concepto folksonomía se refiere al etiquetado de conceptos que se realiza dentro de las comunidades de Internet. Esta visión es capaz de obtener información más precisa de un concepto concreto, una opinión de un objeto tras haberlo usado, mientras que las ontologías tendían a ofrecer información por clases.

En estos últimos años, el uso de la Web ha dado un salto en la forma de en la que los usuarios interactúan unos con otros, originando lo que se conoce como *Web 2.0*. Este fenómeno social facilita ostensiblemente la publicación de experiencias en la Web. La utilización de blogs, wikis, plataformas educativas o redes sociales se ha disparado y ahora nosotros nos proponemos utilizar este conocimiento.

Las experiencias previas de las que se vale un sistema CBR para su ejecución están en forma de *casos* normalmente estructurados. Un problema de la utilización de experiencias de la Web, es que allí, la información no está representeada en estas estructuras. Por lo tanto, se plantea el problema de pasar estas experiencias a casos que pueda tratar una aplicación CBR. Este será el primer problema que empezaremos a tratar en la Sección 2.2.

2.1.5. Aproximaciones previas al WEB-CBR

Antes de comenzar con nuestra aportación, hemos de hablar de trabajos previos al nuestro, de cómo han afrontado los problemas que nosotros trataremos y qué soluciones y resultados han obtenido.

Leake y Powell (2007) proponen la utilización de información web centrándose en el conocimiento de adaptación. Con este objetivo, utilizan tres fuentes web como son *Wikipedia*, *OpenCyc* y *Geonames GIS database*. Combinando información de estos tres sitios, consiguen realizar adaptaciones de casos en el dominio de viajes y turismo. Aseguran desarrollar técnicas de

adaptación independientes del dominio de trabajo. No obstante, una de las fuentes de información que utilizan es Geonames GIS database, una base de datos con más de 8 millones de nombres geográficos. Si para otro dominio no se dispusiera de una fuente de información tan específica, sus técnicas de adaptación podrían no resultar útiles. También cuentan con la suposición de que la información más importante de los artículos de Wikipedia se encuentra en el primer párrafo. Con esto y con otras páginas que enlazan a cada artículo, realizan técnicas de filtrado para eliminar elementos que no sean relevantes y trabajar así con Wikipedia a modo de ontología, destacando que los enlaces de Wikipedia contienen ciclos.

La versión posterior [Leake y Powell (2008)], añade perfiles de usuario a su aplicación. Divide el conocimiento en cuatro fuentes principales: preferencias del usuario, construidas mediante las *feedbacks* del usuario en adaptaciones; conocimiento del dominio, construido guardando información de búsquedas anteriores; perfiles de las fuentes web, construyendo estadísticas sobre a cuándo y a qué fuente se accede, y una base de casos de adaptaciones previas. Al igual que en la versión anterior, se incluyen una serie de reglas para intentar realizar adaptaciones.

Otra aproximación es la descrita en Smyth et al. (2009), que destaca que buena parte de la experiencia de un usuario queda registrada en sus rutas de navegación o en las queries que le han llevado a determinada página. Para poder disponer de este conocimiento, crearon una herramienta integrada en los navegadores web que recopilaba la información necesaria. El usuario debía seleccionar el dominio en el que iba a realizar sus búsquedas y así se creaban repositorios con opiniones, experiencias, votaciones, ... Posteriormente, por supuesto, se necesitaban técnicas de eliminación de ruido.

2.1.6. CBR Multiagente

Una de nuestras ideas será el empleo de la información de Internet como un medio de obtención de casos. Por lo tanto, podríamos decir que aparte de la base de casos local empleada en las aproximaciones CBR clásicas, dispondremos de otra base de casos online. Al hablar de sistemas CBR con varias bases de casos, hemos de mencionar brevemente los sistemas CBR distribuidos. Se basan en la distribución de recursos para obtener una mejora en el rendimiento de los sistemas CBR. En Plaza y McGinty (2005), encontramos una clasificación de los sistemas CBR atendiendo al número de bases de casos que manejan (cómo está organizado el conocimiento) y al número de agentes que trabajan en los mismos (cómo está procesado el conocimiento del sistema). En la figura 2.3 encontramos sistemas con uno o múltiples agentes, y con una o varias bases de casos. Los sistemas de único agente y una sola base de casos, son los CBR clásicos.

Las otras tres alternativas que quedan en la Figura 2.3 equivalen a los sistemas distribuidos. Encontramos:

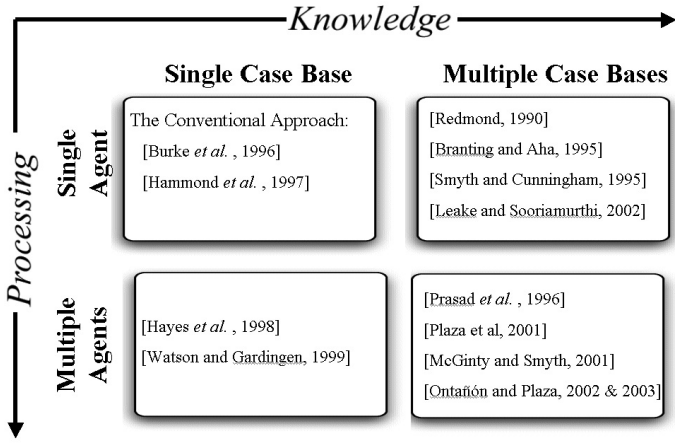

Figura 2.3: Distribuciones de sistemas CBR según número de agentes y número de bases de casos. Figura obtenida de Plaza y McGinty (2005).

- Agente único y varias bases de casos: Son aproximaciones en las que un único agente obtiene experiencias de múltiples bases de casos. Para tratar esta situación, se precisa de *Multi Case-Based Reasoning* y se deben tratar problemas como la mezcla de casos de diferentes bases o cuándo recurrir a una sola de las bases de casos.

- Varios agentes y una sola base de casos: Este enfoque se basa en la cooperación de varios agentes que extraen información de la misma base de casos.

- Varios agentes y varias bases de casos: En esta opción, a diferencia de la anterior, los agentes gozan de mayor *autonomía*, para desempeñar sus tareas. Un agente ha de determinar cuándo una tarea le compete o no y debe ser capaz de poder interactuar con el resto de los agentes [Prasad (2000)].

Nuestro proyecto está orientado al uso de un único agente para después variar entre utilizar una o dos bases de casos. Comenzaremos con un modelo en el que todos los casos son obtenidos de la Web, y a continuación, pasaremos a emplear tanto base de casos online como local, pero siempre con un único agente. Todo esto se explicará en la siguiente Sección.

En Leake y Sooriamurthi (2001) se estudian los problemas y las ventajas de utilizar más de una base de casos. Explica que el utilizar una segunda base de casos puede dar lugar a problemas de disponibilidad, como problemas de permisos; problemas de eficiencia, como solapamiento en la recuperación o problemas de espacio; problemas de mantenimiento al combinar y estandarizar casos de ambas bases, que podría suponer la pérdida de las actualiza-

ciones que recibiría una base de casos en solitario, ... Sin embargo, también propone una serie de ventajas que conllevaría trabajar con dos bases de casos. La ventaja más inmediata es la disposición de más casos para recuperar. Destaca que si las bases de casos son diferentes en cuanto a semántica o gestión de los domnios de trabajo, ... se puede dar lugar a inferencias que no pueden alcanzarse por medio de cada una de las bases en solitario. Explica también que las descripciones de la base de casos o de la recopilación de los casos en la construcción de la base puededeterminar la aplicabilidad de los casos. Luego enumera una serie de análisis comparativos que se pueden conseguir entre varias bases de casos.

2.2. Modelo teórico para WEB-CBR

Nuestro objetivo es la integración de conocimiento online en un sistema CBR. En esta Sección estudiaremos los aspectos que se deberían modificar con respecto a lo estudiado en la Sección 2.1. Aquí se explicó que el conocimiento manejado por los sistemas CBR no se limitaba únicamente al mantenimiento de la base de casos. Existía más conocimiento, que además lo agrupamos en los denominados *knowledge containers*. Vamos a estudiar qué supone la inclusión de conocimiento online en estos *kc* en cuanto a modificaciones estructurales y a modificaciones funcionales.

En esta Sección no nos centraremos en la manera de obtener el conocimiento de la Web, pues eso es algo que dependerá concretamente de la fuente web y que se estudiará posteriormente en el marco práctico (Capítulo 4). Supondremos que disponemos de la información online necesaria para la creación de este modelo en el formato preciso para su integración.

El estudio aquí propuesto dio pie a un artículo enviado y aceptado para el ICCBR [Recio-García et al. (2010)].

2.2.1. Aspectos estructurales

Comenzaremos estudiando los aspectos estructurales a modificar de los sistemas CBR. Aquí vamos a considerar la utilización de la Web como medio de obtención de casos. Recordamos que el *knowledge container* kc_{cb} era el encargado del almacenamiento de los casos y será el relevante en esta Subsección.

El empleo de casos obtenidos desde la Web lo dividiremos en dos opciones: primero consideraremos los casos online como una única base de casos y después, estudiaremos una posible combinación entre base de casos local y base de casos online.

2.2.1.1. Base de casos única

A simple vista, la inclusión de conocimiento de una fuente web puede realizarse de diferentes maneras. Por ejemplo, antes de comenzar cada ciclo CBR, podríamos tratar de conectar con una base online para intentar *llenar* el *knowledge container* adecuado de la aplicación. Las modificaciones que haríamos en el ciclo CBR son sencillas. Únicamente haría falta una fase de que denominaríamos *update* que consistiría en conectar y descargar la información necesaria. Esta información se guardaría, en principio en kc_{cb}, sin embargo, para evitar confusiones con la siguiente aproximación diremos que se almacena en kc_{wcb}, *Knowledge Container for Web Case Base*. Una vez realizada la carga de los casos, se continuaría la ejecución del ciclo CBR con total normalidad.

Por tanto, a nivel práctico, necesitamos un método o algoritmo que recupere los casos de la fuente online. Consideramos que debido al gran tamaño de las fuentes de información en Internet, la descarga de casos se hará a partir de la consulta entrante. Con esto, queremos indicar que no se hará una descarga de todos los casos antes de la primera ejecución. Habrá una fase *update* en cada ejecución del ciclo. Este método lo denominaremos $WebRetrieval(Q)$, donde Q es la consulta de usuario. Se pueden apreciar las modificaciones en la Figura 2.4.

2.2.1.2. Base de casos mixta

Con lo explicado en la Sección anterior, estaríamos logrando crear una herramienta que dependiera únicamente una base de casos Web. Sin embargo, nuestro verdadero objetivo, sería poder combinar casos almacenados localmente con los que recuperaríamos de la Web. Así, disminuiríamos tiempos de carga, pues habría ocasiones en las que no sería necesario acudir a la base casos Web. Con esto, podríamos tener otra aproximación que, del mismo modo que antes, previamente a la ejecución del ciclo CBR conectara vía web a una base externa y completara la información que posee de manera local. Ahora, la fase *update* se encargaría de rellenar el contenedor kc_{wcb} con los nuevos casos descargados, pero a la vez, dispondríamos de el kc_{cb} con los casos de la base local.

Estamos optando por diferenciar los casos obtenidos via online de los ya previamente almacenados en local, una buena opción para evitar incrementar en exceso el tamaño o añadir ruido a la base de casos local, que conllevaría a pérdida de precisión. Disponemos de casos recuperados de la base local y casos recuperados de la base web y debemos emplearlos adecuadamente. Con el fin de acelerar ejecuciones, podemos optar por la opción de recurrir a la fuente online cuando los casos recuperados de la base local no sean *lo suficientemente buenos*, tal y como planteamos a continuación.

Por tanto, vamos a necesitar construir un método $CombinedRetrieval(Q)$,

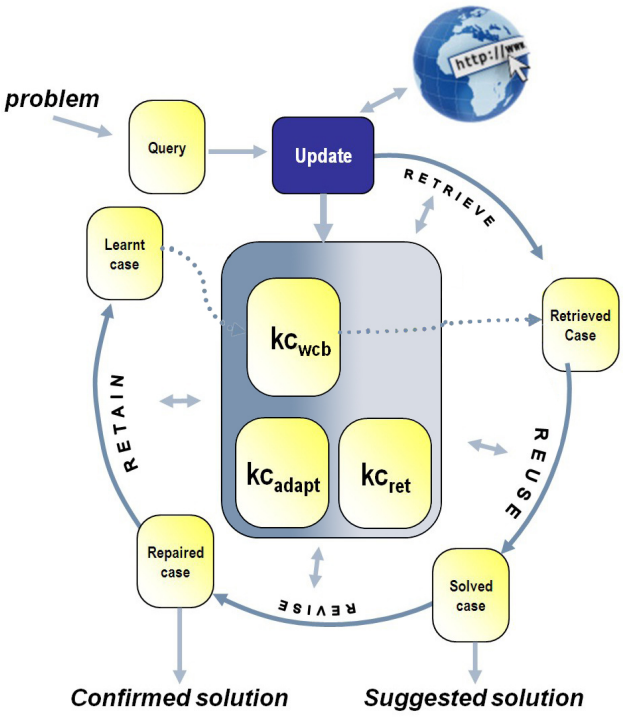

Figura 2.4: Ciclo CBR para base de casos web única.

donde Q vuelve a ser la consulta de usuario, que tras la llegada de Q recupere los casos más similares de la base de casos local. Si los casos no satisfacen las exigencias necesarias, se procederá a ejecutar la fase de *update* y rellenar el kc_{wcb}. A continuación se procede a la recuperación de los casos más similares a Q de este último *knowledge container*. En la Figura 2.5, podemos apreciar cómo el caso recuperado, *Retrieved case*, puede provenir ahora tanto de kc_{cb} como de kc_{wcb}. Hay que destacar que los casos recuperados de kc_{wcb} no tiene por qué ser mejores que los de la base local. Por tanto, hay que decidir cómo actuar ante esta situación. Se podría decidir entre hacer la unión de casos recuperados de ambas bases, por ejemplo obtener los k casos con mayor medida de similitud de entre ambas opciones. Existen diversos estudios que tratan de decidir de manera automática de qué base de casos recuperar la información. Como ejemplo, podemos destacar a Leake y Sooriamurthi (2002). Los autores proponen métodos para que un sistema Multi-CBR pueda elegir automáticamente entre diferentes bases de casos, dando razones por las que una base de casos debe ser empleada. Miden diferencias entre la representación de los casos de una base y de otra y se apoyan en un valor umbral para la toma de decisión.

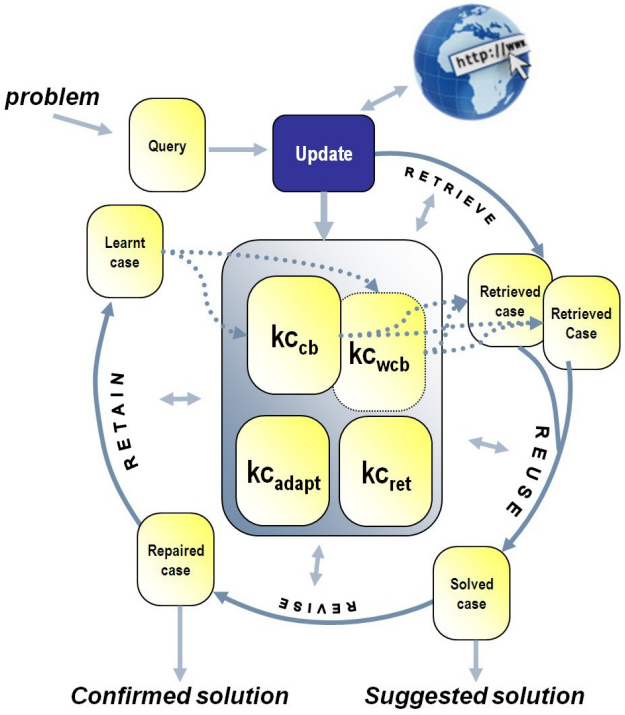

Figura 2.5: Ciclo CBR para modelo mixto de base de casos.

Es muy probable que los casos de ambas bases de casos[2] tengan formatos o estructuras diferentes, sobre todo si la creación de la base local es anterior a la integración de la base online. Esto implica un cambio a la hora de trabajar con las funciones de similitud características del Razonamiento Basado en Casos. Una opción, que es por la que hemos optado al llevarlo a la práctica, consiste en fusionar las estructuras de ambas representaciones de los casos. Así, el cálculo de medidas de similitud o aplicación de funciones de adaptación se hará de la misma manera, independientemente del origen del caso recuperado.

Destacar también que utilizar la misma organización de la base de casos tanto para kc_{cb} como para kc_{wcb}, simplifica y facilita la recuperación de casos y el empleo de las funciones de similitud.

2.2.2. Aspectos funcionales

Como indicábamos al introducir los *knowledge containers* en la Sección 2.1.2, la base de casos no es el único conocimiento necesario para la ejecución

[2]Hablamos continuamente de dos bases de casos, local y online, pero perfectamente sería aplicable a más de dos.

de una aplicación CBR. Por lo tanto, si en el apartado anterior, estudiamos de manera teórica qué supondría la integración de una fuente Web para obtener casos (kc_{wcb}), ahora veremos cómo utilizar este conocimiento para mejorar o ampliar kc_{ret} o kc_{adapt}. Esta situación dará pie a una serie de modificaciones funcionales que estudiamos a continuación.

2.2.2.1. Recuperación

En nuestro afán de seguir mejorando la usabilidad de la herramienta, nos damos cuenta de que las consultas a bases externas conllevan más tiempo del que se utiliza para recuperar casos locales. De inmediato llegamos a otra modificación. Ejecutaremos la *recuperación web* únicamente cuando sea necesaria, obteniendo un ahorro de tiempo considerable. En la Sección 2.2.1.2, hemos introducido ésto, dejando como objetivo la implementación de un algoritmo $CombinedRetrieval(Q)$. Este algoritmo recupera casos de la base local y decide si son *lo suficientemente buenos*, o si por el contrario, hace falta recurrir a otra base de casos. Puede haber dos razones, por tanto, por las que se precise conocimiento de fuente externa:

- El máximo valor de similitud devuelto por los casos locales no supera un determinado valor umbral.
- La solución propuesta al usuario no satisface sus exigencias.

En ambos casos, se precisan restructuraciones en el ciclo CBR. Para el primer caso habría que fijar un valor numérico umbral. Tras la ejecución de la primera fase del ciclo clásico del CBR, la de recuperar los casos más similares a la consulta, se comprueba si el valor de similitud es mayor que el umbral. De no serlo, se intentaría rellenar el kc_{wcb} vía online y se volvería a ejecutar la etapa de recuperación.

Para la segunda alternativa, suponemos una ejecución correcta del ciclo CBR hasta la etapa de revisión. En este momento, se le proponen al usuario las soluciones más semejantes a la consulta solicitada. Si ninguna de las soluciones son de su agrado, pedirá a nuestra herramienta otras posibles soluciones, para lo cual, volveremos a la fase de recuperación y cargaremos información de nuestra base web. Posteriormente continuaremos ejecutando el ciclo CBR hasta el final. Es posible que tras la ejecución de esta *segunda vuelta* no se encuentren resultados mejores a los obtenidos de forma local. Esto dependerá de la calidad de la base web que estemos empleando.

Una vez finalizada la ejecución, suponemos que si la solución ha sido lo suficientemente buena como para satisfacer al usuario, merecería la pena guardarla en la base de casos local para futuras consultas. También podríamos optar por descartarla directamente o incluso decidir si almacenarla o no dependiendo de una *medida de calidad* que impongamos nosotros. Deja-

Modificación deseada	Algoritmo necesario
Base de casos única	WebRetrieval
Base de casos mixta Recuperación	CombinedRetrieval
Adaptación	WebAdaptation

Tabla 2.1: Modificaciones y algoritmos.

mos, por tanto, abierta la posibilidad de elegir cualquier tipo de política de actuación para este tema [Scime (2005)].

2.2.2.2. Adaptación

En la Sección anterior, hemos explicado cómo completar la información almacenada en el kc_{cb}. Lo que proponemos a continuación es aprovechar también el conocimiento online para mejorar otros *knowledge containers* enumerados en la Sección 2.1.2. Concretamente, nos centraremos en los contenedores que, con anterioridad, denominamos kc_{adapt} y kc_{ret}, los contenedores referentes al conocimiento de adaptación y de recuperación respectivamente.

Cuando hablamos de kc_{adapt}, nos estamos refiriendo a un conjunto de reglas o pasos para transformar un caso almacenado en nuestra base en una solución que consideremos válida y que se asemeje todavía más a la consulta solicitada. Si consideramos la consulta o *query* como un conjunto de pares (atributo, valor): $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$, una posible adaptación sería la susutitución de alguno los valores de uno de sus atributos v_i. También puede optarse por otros tipos de adaptación más complejos, que requieran la sustitución de un mayor número de atributos e incluso la omisión de alguno de ellos o la inclusión de alguno nuevo. Este tema es algo más complejo de enfocar a nivel teórico. Podríamos desrrollar técnicas de adaptación automática, métodos que indiquen cómo proceder a la hora de aplicar una sustitución. Sin embargo, en este punto es difícil generalizar, debido a que estas sustituciones de atributos dependen en gran medida del dominio en el que nos movamos. Así, en Secciones posteriores (véase Sección 4.3.3), se ha desarrollado un algoritmo de adaptación que obtiene conocimiento vía Web, pero específico para el ámbito en el que se trabaja. Queda por tanto pendiente la futura implementación de un algoritmo $WebAdaptation$. La Tabla 2.1 resume las modificaciones y los algoritmos a

2.2.3. Gestión y mantenimiento de la base de casos

En este apartado vamos a estudiar el comportamiento del modelo híbrido que estamos desarrollando para unos casos extremos iniciales de la base local. Consideraremos la opción de encontrarnos la base de casos local totalmente vacía o, el extremo opuesto, hallarnos una base local tan completa que cubra

todo el posible espacio de búsqueda.

- Estudiando la primera alternativa, nos encontramos un caso ya propuesto por Leake y Sooriamurthi (2003). Nos hallamos ante la necesidad de operar únicamente con la base web. Este proceso nos llevaría inicialmente a una *importación* de casos desde la red a nuestra base local. Esta situación es bastante común cuando queremos desarrollar una nueva aplicación CBR y no hemos podido conseguir una base de casos.
- Mirando el lado opuesto, podríamos encontrarnos con una base de casos muy completa, que tras muchas ejecuciones ha llegado a un estado en el que todos sus casos ocupan un sector muy concentrado del espacio de soluciones. Podría ser ésta una situación en la que todas las soluciones que devuelva el programa para las consultas del usuario estén ubicadas en esta zona. La incorporación de una nueva base de casos podría dar lugar a aumentar la diversidad. Se podría dar la situación en la que se recuperaran casos que sería imposible inferirlos mediante adaptaciones de los que había en la base local.

Los anteriores estados de la base de casos pueden llevarnos a una pregunta. Partiendo de una base de casos local y añadiendo una base de casos online, suponemos que el espacio de soluciones cubierto por nuestra aplicación comenzará a aumentar mientras almacena nuevas experiencias. ¿Sería posible que, tras muchas ejecuciones de este modelo mixto de CBR, volvieramos a la situación en la que ya no se avance más en la expansión del espacio comprendido por la base de casos? Cabría esperar que sí, pero una de las características de las fuentes de conocimiento online es su constante y veloz crecimiento, sobre todo si nos referimos a bases de casos que pueden ser alteradas por las experiencias de miles de usuarios. Además, siempre se podría realizar la integración de una nueva fuente web, que abriría de nuevo las posibilidades de aumentar la diversidad de las soluciones.

2.3. Conclusiones

En este Capítulo se ha presentado brevemente el Razonamiento Basado en Casos, haciendo hincapié en aquellas facetas que se verán afectadas en la inclusión de conocimiento vía Web. Hemos presentado un modelo teórico de WEB-CBR, dónde hemos descrito aquellos aspectos tanto estructurales como funcionales que habrá que modificar. Se han dejado en el aire, tres algoritmos que se deberán implementar para conseguir realizar las modificaciones deseadas y que serán detallados en futuros Capítulos. Estos algoritmos dependerán de la fuente de la que podamos obtener la información requerida. Por consiguiente, el siguiente paso es buscar una fuente de conocimiento online que podamos integrar en nuestro modelo teórico de CBR.

Capítulo 3

Fuentes Web

Resumen: En este Capítulo presentaremos un estudio de diferentes fuentes web para expoltar en nuestro proyecto. Explicaremos una serie de ventajas y desventajas de cada una de ellas hasta decidirnos por la mejor opción. A continuación, expondremos la adaptación de esta base de datos externa con la arquitectura de jCOLIBRI2.

3.1. Introducción

Ahora que hemos expuesto el marco teórico de nuestro estudio indicando cómo esperamos que se comporte el ciclo CBR, llega el momento de ponerlo en práctica. Para ello, el primer paso es encontrar una fuente de conocimiento externa que podamos integrar en la arquitectura de jCOLIBRI2. La primera alternativa, y la que se venía usando con frecuencia era utilizar técnicas de Procesamiento del Lenguaje y Extracción de Información sobre el contenido textual de la web. Se pretende obtener información estructurada de fuentes no estructuradas. Se emplean técnicas como análisis morfológico, sintáctico o semántico. Estas técnicas son complicadas de utilizar, y más aún en la web. A los habituales problemas de ambigüedades tanto léxicas, p.ej. palabras polisémicas, como estructurales a la hora de construir los árboles sintácticos con problemas de dependencias entre sintagmas o tratamiento de negaciones (que depende de cada idioma), se unen los inconvenientes de los errores gramáticales, ortográficos o abreviaciones empleadas por la gran mayoría de los internautas.

Compañias como Google o Yahoo! comenzaron a desarrollar herramientas en las que ofrecen la información de manera ya estructurada. Las ventajas que nos ofrecen son inmejorables para evitar algunos de los problemas expuestos. Ahora es más fácil acceder a los dominios en los que uno está interesado y extraer la información de los campos que requiera. Podemos imaginar una

fuente web estructurada como una tabla en la que uno puede solicitar la información que necesite de cada una de las columnas en las que se divide cada registro o instancia. Así, podríamos preguntar *quién escribió tal libro* o *quién dirigió tal película* consultando los campos *autor* o *director* que podrían tener perfectamente las tablas sobre los libros o películas que buscamos.

3.2. Estudio de diferentes fuentes Web

Una vez expuestas las posibles ventajas que esperamos encontrar explotando estas nuevas fuentes de información, pasaremos a estudiar cuál o cuáles de ellas son interesantes para incorporar a nuestro proyecto. Hemos investigado las cuatro fuentes más representativas que hemos encontrado: Google Squared, Yahoo! Query Language, Search Monkey y Freebase.

3.2.1. Google Squared

Google Squared, lanzado en Junio de 2009, es una herramienta que pretende cambiar radicalmente el rumbo en lo que a búsqueda web se refiere. Con esta aplicación, Google escanea la web en busca de de la información solicitada por el usuario, pero además muestra los resultados en forma de tabla. Cada registro de la tabla se conoce como hecho o *fact*, y cada columna representa sus atributos, también denominados *squares*. Estos *squares* pueden personalizarse para mostrar los que el usuario desee. Así, si estamos preguntando sobre un tema como "presidentes de Estados Unidos", podremos añadir columnas como años de servicio o religión a las que ya vienen por defecto como fecha de nacimiento, vicepresidente, ... Una vez construida la tabla con las columnas deseadas, podremos exportarla en forma de archivo CSV.

Desde el punto de vista del Razonamiento Basado en Casos, podemos considerar cada uno de los registros de la tabla como un caso, y cada una de las columnas como los atributos de dicho caso. El gran problema que encontramos para esta alternativa es que carece de una API[1] que nos ayude a obtener la información de manera inmediata desde la plataforma de jCOLIBRI2, que es donde queremos incluir esta nueva funcionalidad. Otro problema a la hora de emplear Google Squared en una herramienta CBR es que carecemos de un mapa que nos indique qué atributos son posibles construir para cada *square*. Éste, en realidad, no es un problema grave, sino un inconveniente temporal, que nos obliga a realizar un pequeño estudio previo del dominio buscando qué posibles columnas podrían añadirse, aunque bien es cierto, que Google ofrece sugerencias.

Hemos observado que es posible especificar por medio de la dirección url el *square* buscado, pero desde aquí no podemos indicar qué columnas queremos

[1] Application Programming Interface

que nos muestre, cosa de vital importancia debido a que estamos hablando de los atributos de un caso del CBR. Ésto se debe a que la herramienta trabaja con AJAX. Esta tecnología, empleada para acelerar la navegación web, no precisa de la recarga de la página web que muestra para modificarla, por lo que la dirección url no varía.

3.2.2. Yahoo! Query Language

Yahoo! Query Language (YQL) es un lenguaje similar a SQL que permite consultas, filtrados y uniones de datos a través de servicios web. El objetivo de esta plataforma es la recolección de información de diferentes sitios web para que pueda ser consultada a través de su API. Así, no sería necesario el deber documentarse sobre la forma de acceder a cada servicio web de cada página de la que queremos descargar datos. El modo de utilizar YQL está basado en tablas, de modo similar a consultas SQL. A simple vista, una posible integración de esta herramienta en jCOLIBRI2 no se antoja excesivamente complicada. El problema viene al fijarnos en los datos que podemos manejar. Debemos adaptarnos a las tablas proporcionadas por YQL, que aunque bien es cierto que poco a poco irán creciendo (sobre todo gracias a los aportes de comunidades de usuarios), actualmente son pocas o con pocas propiedades por las que preguntar. Destacar también que las tablas construidas por Yahoo! son mucho más completas que las construidas por las aportaciones de usuarios, pero el número es bajo, 112 tablas, un 15 % del total.

3.2.3. SearchMonkey

SearchMonkey es una iniciativa de Yahoo! que propone a los desarrolladores de sitios web utilizar datos estructurados para hacer los resultados de búsqueda más útiles y dirigir de manera más relevante el tráfico a sus sitios. Solicitan a los diseñadores de sitios web que incluyan unas líneas de código en sus páginas. Gracias a esto, SearchMonkey recolectará información, la estructurará y la pondrá a disposición de los usuarios a través del buscador de Yahoo!. Si ahora nos dirigimos a la página principal de Yahoo![2] y buscamos información sobre restaurantes en Nueva York, los primeros resultados aparecerán de manera estructurada, pudiendo acceder directamente a fotos, críticas o ubicaciones de estos lugares.

Un gran inconveniente es que no hemos encontrado ninguna forma de acceder a esta información. Es posible acceder a la base de datos donde se SearchMonkey recopila la información por medio de una herramienta online que ellos mismos proporcionan, pero su funcionalidad es bastante limitada e impide realizar consultas de una manera general.

[2] http://www.yahoo.com

3.2.4. Freebase

Freebase es un repositorio abierto de información estructurada. Gracias a la interfaz ofrecida a los usuarios para que añadan «conceptos» a la base de datos, actualmente se albergan más de 12 millones de instancias, con más de 3.000 tablas y más de 30.000 propiedades. Freebase se organiza en *domains*, *types*, *properties* y *topics*.

- Los *topics* son cada uno de los conceptos o instancias únicos de Freebase. Por ejemplo, un *topic* puede ser una película o un plato culinario.
- Los *topics* se organizan en diferentes tipos o *types*. Cada concepto puede pertenecer a uno varios *types*. Por ejemplo, Leornardo da Vinci pertenecerá a varios tipos como pintor, escultor, inventor, ingeniero, ...
- Las propiedades son los campos por los que se puede preguntar sobre cada concepto dentro de cada tipo. El tipo *Company* tiene propiedades como empleados, fundador, socios, ...
- Los dominios, simplemente, son una manera de organizar los *types* de un modo más general. Encontramos dominios como música, cine, comida, ...

Para acceder a la información de Freebase, se dispone de un lenguaje llamado MQL (Metaweb Query Language), similar a SQL. MQL puede ser ejecutado a través de una API muy sencilla de utilizar, y a mediante objetos JSON podemos recibir información en diferentes lenguajes de programación como Java, C, C++, ...

3.3. Discusión: ventajas y desventajas

Llega el momento de discutir cuál de las anteriores alternativas es la que más nos conviene para incluir en la plataforma de jCOLIBRI2.

- Optamos por descartar Yahoo! Query Language, que aunque el modo de utilizarlo a través de su API es sencillo, la información que albergan es bastante limitada a los pocos dominios que ofrecen.
- La opción de SearchMonkey la rechazamos también de manera inmediata ya que la carencia de una API pública.
- Google Squared sería la mejor opción. Busca por toda la red y recopila la información de manera estructurada. Sin embargo, la falta de una API es un impedimento demasiado grande. Si en algún momento se publicara una API, ésta sería sin duda la primera opción a tener en cuenta.

- Por tanto, escogemos Freebase. Es cierto, que tenemos que adaptarnos a la información que hay almacenada en esta base de datos, pero cuenta con tantos conceptos (12 millones) y suficientes dominios, que se ajusta lo suficiente a nuestro estudio. La integración de esta plataforma en jCOLIBRI2 (que se describirá en la Sección 4.4) es asumible gracias a la API ofrecida.

3.3.1. La Organización de Freebase

A continuación, vamos a explicar un poco más detenidamente la estructura y funcionamiento de Freebase.

Existen 75 dominios en Freebase en los que se organiza toda la información. Cada dominio es precedido por el símbolo '/', p.ej. /education o /government. Para acceder a cada tipo ubicado dentro de cada dominio, lo único que hay que hacer es concatenar el nombre del tipo al del dominio. Así, si queremos acceder al *type* actor de una película, deberemos concatenar el dominio *film* con el tipo *actor*: /film/actor. Del mismo modo, las propiedades de cada tipo se concatenan a éste, obteniendo por ejemplo /film/actor/nacionality. Para facilitar la búsqueda de propiedades y tipos dentro de la estructura de Freebase, disponemos de la herramienta «Schema Explorer»[3].

Para poder hacer consultas dentro de esta base de datos, Freebase ofrece un lenguaje denominado MQL (Metaweb Query Language). Consiste en crear una estructura de campos-valores dejando el campo desconocido como null o []. Por ejemplo, si quisiéramos mirar quién dirigió la película «Infiltrados» realizaríamos la siguiente consulta:

```
[{
  "id" : "/en/the_departed",
  "/film/film/directed_by" : null
}]
```

id es el identificador del concepto y /film/film/directed_by corresponde a la propiedad director, dentro del tipo film y dentro del dominio film. Si ejecutamos esta consulta en el editor online que nos ofrece Freebase[4], obtendremos el siguiente resultado:

```
{
  "code":          "/api/status/ok",
  "result": [{
    "/film/film/directed_by": "Martin Scorsese",
    "id":             "/en/the_departed"
  }],
```

[3] http://schemas.freebaseapps.com/
[4] http://www.freebase.com/app/queryeditor

```
  "status":        "200 OK",
  "transaction_id": "cache;cache01.p01.sjc1:8101;2010-05-26T10"
}
```

Ahora vemos cómo el campo /film/film/directed_by ha sido rellenado con "Martin Scorsese". De esta manera podemos preguntar por las propiedades de cualquier *topic* de Freebase. Los resultados también pueden venir en forma de lista, lo que implicará que el campo buscado deberemos dejarlo como [] en vez de null. Vamos a buscar todas las películas dirigidas por Francis Ford Coppola:

```
[{
  "id" : "/en/francis_ford_coppola",
  "/film/director/film":[]
}]
```

Obtnemos una lista:

```
{
  "code":          "/api/status/ok",
  "result": [{
    "/film/director/film": [
      "Apocalypse Now",
      "Bram Stoker's Dracula",
      "Dementia 13",
      "New York Stories",
      "One from the Heart",
      "Rumble Fish",
      "The Conversation",
      "The Godfather Part II",
      "The Godfather Part III",
      "The Godfather Saga",
      "The Outsiders",
      "The Rainmaker",
      "Tucker: The Man and His Dream",
      "The Cotton Club",
      "The Godfather",
      "Gardens of Stone",
      "The Rain People",
      "Youth Without Youth",
      "You're a Big Boy Now",
      "The Bellboy and the Playgirls",
      "Peggy Sue Got Married",
      "The Terror",
```

```
      "Jack",
      "Apocalypse Now Redux",
      "Captain EO",
      "Finian's Rainbow",
      "Tetro",
      "Supernova",
      "The Godfather Trilogy: 1901-1980"
    ],
    "id": "/en/francis_ford_coppola"
  }],
  "status":        "200 OK",
  "transaction_id": "cache;cache01.p01.sjc1:8101;2010-05-26T11"
}
```


Desde el punto de vista del Razonamiento Basado en Casos, podríamos considerar cada *topic* como un caso. De esta forma, se podrían recuperar todos los casos pertenecientes a un *type*. Y cada una de las propiedades se considerarían como los atributos de cada caso. Por ejemplo, dentro del tipo libros, podríamos solicitar una lista de libros con atributos como autor, género, idioma original, ... Cada uno de esos libros sería un caso con el que poder ejecutar aplicaciones CBR. Esto nos conduce directamente a plantearnos un algoritmo para automatizar el proceso de extracción de casos desde Freebase. Este algoritmo y otros se especificarán en el siguiente Capítulo.

3.4. Conclusiones

Se ha realizado un estudio de diferentes fuentes web que proporcionan información estructurada y que facilitan considerablemente el trabajo de obtención de conocimiento online. La opción escogida, Freebase, era la que nos parecía más sencilla de integrar en la arquitectura de jCOLIBRI2 y que además poseía una cantidad de información lo suficientemente representativa. Una vez explicado el marco teórico y seleccionada la fuente web que pretendemos explotar, el siguiente paso será llevar a cabo la integración con jCOLIBRI. Para ello, será necesaria la construcción de una manera de conectar con Freebase, para proseguir luego con la implementación de los algoritmos que se introdujeron en la Sección 2.2

Capítulo 4

Marco Práctico

Resumen: En este Capítulo se realizará la integración de conocimiento Web, de Freebase, en el framework jCOLIBRI. Para ello, se expondrá una introducción a dicho framwork, una descripción al dominio de trabajo antes de proponer tres algoritmos y la implementación de la conexión con Freebase para el intercambio de datos.

4.1. Introducción

Una vez enmarcado el modelo teórico a seguir (Sección 2.2) y escogido la fuente Web estructurada con la que trabajaremos (Sección 3.3), llega la hora de llevar a la práctica nuestro proyecto. Para ello, nos proponemos a implementar una integración del conocimiento de Freebase en el framework jCOLIBRI2. Una vez que haya construido dicha integración jCOLIBRI dispondrá de la opción de utilizar Freebase como fuente de conocimiento. Sin embargo, antes de esto se deben resolver los problemas que se dejaron planteados en la Sección 2.2. En dicha Sección, se explicaba las modificaciones estructurales y funcionales que deberían llevarse a cabo para poder incluir información online en un sistema de Razonamiento Basado en Casos. También se indicaron tres algoritmos que deberían implementarse para llevar a cabo dichas modificaciones. Estos algoritmos no fueron implementados en el Capítulo 2 porque considerábamos que dependerían en exceso de la fuente web escogida para obtener información. Por lo tanto, hasta la conclusión del Capítulo 3 no podíamos formularlos.

Entonces, antes de explicar la integración de Freebase en jCOLIBRI2, explicaremos los algoritmos que hemos desarrollado. Para facilitar la comprensión de los algoritmos, realizaremos ejemplos para cada uno de ellos. Uno de los algoritmos, *CombinedRetrieval*, propuesto en la Sección 2.2.2.1, utilizaba dos bases de casos, local y online. Por tanto, para su implementación

necesitamos disponer de una base de casos local. De ahí la decisión de utilizar el dominio de trabajo explicado a continuación.

4.2. Dominio de trabajo

Para mejorar el entendimiento de los algoritmos que se desarrollarán en la Sección 4.3, se desarrollaran una serie de ejemplos. Como acabamos de explicar, necesitamos una base web y una base local para trabajar, por lo menos para implementar *CombinedRetrieval* (Sección 2.2.2.1). Por supuesto, ambas bases de casos han de trabajar en el mismo dominio. Esto nos llevo a la elección del campo de las recetas de cocina.

Este dominio es el que se emplea en el Computer Cooking Contest, un concurso de Razonamiento Basado en Casos que trabaja con recetas de cocina y al que el grupo GAIA ha presentado tres trabajos en los últimos dos años [DeMiguel et al. (2008), Herrera et al. (2008)]. Esto, unido a que en Freebase existe un dominio llamado *Food & Drink*, nos condujo a decantarnos por esta opción. Ahora quedaba seleccionar una base de casos de una de las participaciones en el Cooking Contest y estudiar el dominio de Freebase para ver cómo buscar la información requerida.

4.2.1. Computer Cooking Contest

Vamos a explicar brevemente en qué consiste el Computer Cooking Contest para que el lector entienda qué es lo que se pide a los trabajos enviados y pueda hacerse una idea de qué tipo de conocimiento pueden manejar.

El Computer Cooking Contest es un concurso que se encuentra dentro del ICCBR (International Conference on Case-Based Reasoning). El Computer Cooking Contest (CCC), cuya primera edición fue en 2008, reta a estudiantes e investigadores a implementar un sistema de creación de recetas de cocina. Se les otorga un conjunto de recetas, base de casos inicial, y se pide que los sistemas CBR construidos sean capaces de resolver consultas acerca de ingredientes, tipos de plato, dietas o cocina.

Dispone de cuatro modalidades dentro del concurso:

1. *Main Challenge*: Consiste en responder a consultas que requieran la selección y modificación de la receta de un plato. Ejemplo: *Me gustaría pescado, pero evita el limón y otros cítricos.*

2. *Adaptation Challenge*: El objetivo se centra en las adaptaciones. Ejemplo: *Tengo una receta de tarta pero no tengo ni crema ni vainilla. ¿Qué debería hacer?*

3. *Open Challenge*: La única especificación es comenzar con la base de recetas inicial y realizar cualquier cosa, que sorprenda al jurado.

4. *Student Challenge*: Alternativa para estudiantes que se centran únicamente en algún tipo concreto de tarea como puede ser el procesamiento del lenguaje natural de las queries.

En ediciones anteriores, el grupo de investigación GAIA[1] presentó proyectos a esta competición como fueron ColibriCook [DeMiguel et al. (2008)] o JaDaCook [Herrera et al. (2008)]. Gracias a esto, ya disponíamos de herramientas CBR que trabajaban sobre bases de casos locales y que además hubieran utilizado el framework de jCOLIBRI. Esta situación nos ahorraría mucho trabajo a la hora de combinar ambas bases de casos, que es nuestro objetivo, o realizar comparaciones entre nuestro trabajo y versiones que trabajaban sobre local. De entre las dos opciones, nosotros hemos optado por utilizar la base de casos de ColibriCook como base de casos local.

4.2.1.1. ColibriCook

ColibriCook [DeMiguel et al. (2008)] es un sistema CBR que utiliza la arquitectura de jCOLIBRI2 y fue presentado por el grupo GAIA en la primera edición del Computer Cooking Contest en 2008. Esta aplicación trabaja con una ontología, gracias a la cual, crea sus medidas de similitud. Gracias a esta ontología y a la base de recetas que fue proporcionada por el Cooking Contest, la herramienta trabaja con conceptos como ingredientes, tipo de cocina, tipo de plato, tipo de dieta o tipo de ingrediente. De estos cinco, los dos primeros son los que utilizaremos en Secciones posteriores para realizar nuestros experimentos (Capítulo 5).

4.2.2. Food & Drink en Freebase

Una vez elegida la que será nuestra base de casos local, la de ColibriCook, pasamos a estudiar dónde obtener un conocimiento similar en Internet. *Food & Drink* es un dominio de Freebase que incluye gran cantidad de información acerca de casi todo lo referente a comida y bebida. Como todos los dominios de Freebase, se encuentra en constante crecimiento. Dentro de este dominio, podemos encontrar hasta 58 *types* diferentes. Observamos categorías como restaurantes, chefs, vinos, tipos de texturas de queso, ... Nosotros nos centraremos en el tipo *dish* (plato en inglés). Actualmente, hallamos 2.487 instancias de este tipo.

En el tipo dish nos centraremos en tres propiedades:

- **cuisine**: Tipo de cocina a la que pertenece un plato, p.ej italiana, mediterránea, japonesa, ...
- **type_of_dish**: Corresponde al tipo de plato, como postre, primer plato, ensalada, ...

[1] http://gaia.fdi.ucm.es/

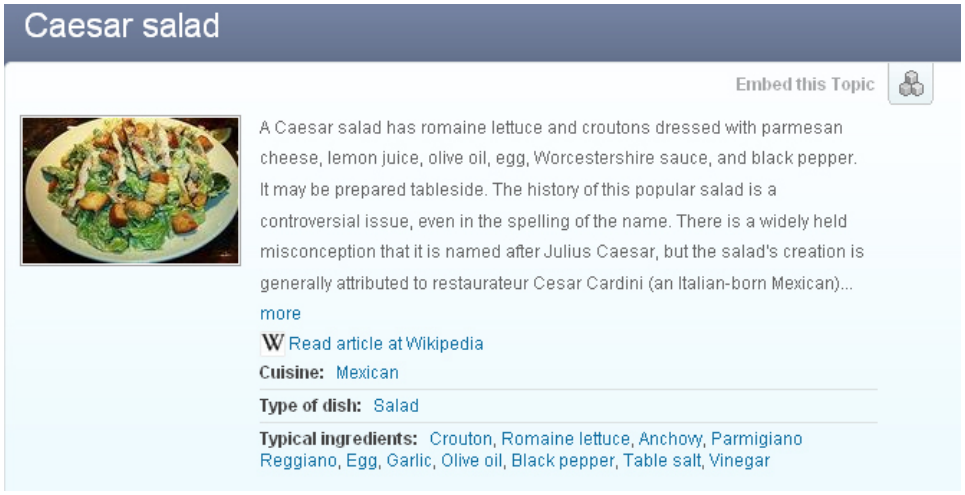

Figura 4.1: Ejemplo de plato de Freebase.

- **ingredients**: Una lista de ingredientes para preparar el plato

Podemos ver uno ejemplos del tipo *dish* en la Figura 4.1. Estas tres propiedades también se encuentran en las recetas de ColibriCook. Por tanto, la elección de este dominio queda suficientemente clara. La recuperación de información de la base de casos local, ya está implementada. Ahora queda ver cómo vamos a obtener los datos necesarios de Freebase. Para ello se dejaron tres algoritmos en la Sección 2.2 que se deberían implementar y que detallamos en la siguiente Sección.

4.3. Algoritmos

Ya hemos visto cuál es el dominio de Freebase, y el *type* dentro de éste al que nos tenemos que dirigir. Ahora queda completar las modificaciones estructurales y funcionales expuestas en el Marco Teórico de nuestro trabajo. Se quedaron por formular tres algoritmos. Para el modelo propuesto en la Sección 2.2.1.1, que trabajaba con una única base de casos, la base de casos web, se dejó pendiente de implementar el algoritmo *WebRetrieval*, que se encargaría de recuperar de la Red los casos necesarios para ejecutar el ciclo CBR.

A continuación, en la Sección 2.2.1.2, se proponía un modelo híbrido que trabajara con base de casos online y local. Complementándolo con lo explicado en el Apartado 2.2.2.1, se precisaba un método que en su momento denominamos *CombinedRetrieval*, que combinara la obtención de casos de ambas bases, y no sólo eso, sino que también decidiera cuando utilizar una base o las dos, con el objetivo de reducir tiempos de ejecución.

Por último, en la Sección 2.2.2.2, se propone la opción de utilizar el conocimiento proveniente de la Red para realizar trabajos de adaptación de casos. Se propone la creación de un algoritmo *WebAdaptation*, que se valga de la información albergada en Freebase para nuestro caso.

4.3.1. WebRetrieval

Presentamos un algoritmo que dada una consulta de usuario, identifique en Freebase el *type* que mejor encaje con los atributos de la query para posteriormente recuperar todos los casos posibles de Freebase con el fin de poder ejecutar el ciclo CBR sobre dichos casos. Esta primera aproximación emplea únicamente la base de casos de Freebase. Como se explicó en la Sección 3.3, los *topics* de Freebase son considerados como los casos de un sistema CBR y como podemos solicitar una lista de todos los *topics* pertenecientes a un *type*, establecemos una relación de $type = Base\ de\ Casos$. Dentro de cada *type* de Freebase, podemos preguntar por el valor de cada una de las propiedades que tiene asociadas para cada *topic* o caso. Con ello, establecemos $Propiedad\ de\ Freebase = Atributo\ del\ Caso$.

Realizaremos una partición del espacio de búsqueda según la query del usuario. Esto se debe a que la API de Freebase devuelve búsquedas excactas y por tanto únicamente podremos recuperar aquellos casos cuyos atributos sean exactamente los que solicita el usuario. Con el fin de obtener un conjunto de casos más amplio sobre el que razonar, haremos sucesivas consultas preguntando en cada una de ellas por un único atributo. Luego haremos la unión de conjuntos para construir el conjunto final, denominado RS_{Web}.

Algoritmo **WebRetrieval(Q)**:

1. Dada una query de entrada $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ compuesta de pares ⟨atributo, valor⟩.

2. Sea FB_{types} el conjunto de types disponibles en Freebase, y $Atr(type)$ el conjunto de atributos (o propiedades de Freebase) asociados con un type de Freebase t_i.

3. Sea $Atr(Q) = \{a_1, a_2, \ldots, a_n\}$ el conjunto de atributos de la query del usuario Q.

4. Encontrar el type de Freebase que contenga el máximo número de atributos en la query:

$$Type_{max} = \{t_i \in FB_{types} \mid |Atr(type_i) \cap Atr(Q)| > |Atr(type_j) \cap Atr(Q)|\,, \\ \forall t_j \in FB_{types} \wedge t_i \neq t_j\}$$

5. Sea M el conjunto de atributos (o propiedades) que $Type_{max}$ y Q tienen en común, con $|M| = m$.

$$M = |Atr(type_{max}) \cap Atr(Q)|$$

6. Incrementalmente, consturir Web queries para Freebase utilizando los atributos de M:
   Sea B_j el conjunto de casos recuperados de Freebase al consultar únicamente utilizando el atributo j-ésimo de los m atributos de M. Por ejemplo, B_3 es el conjunto de casos recuperados cuando consultamos únicamente empleando el atributo a_3.

7. Finalmente, el conjunto recuperado RS_{Web} es:

$$RS_{Web} = \bigcup_{j=1}^{m} B_j$$

8. Intentar rellenar los valores de los atributos restantes $\{Atr(Q) - Atr(M)\}$ para cada caso en RS_{Web}.

El último punto, dice claramente «intentar rellenar los valores de los atributos restantes». Muchos de los casos obtenidos de Freebase están incompletos, tienen atributos sin rellenar. Ésta es una de las principales características de las fuentes de conocimiento online: *la incompletitud de la información*. Lógicamente, es prácticamente imposible mantener una base de datos online pidiendo a los usuarios proporcionar toda la información necesaria porque acabarían por no incluir sus aportaciones. Por supuesto, se podría intentar obtener la información restante de otras fuentes web, pero es algo que no hemos incluido en nuestro trabajo.

Con este algoritmo dispondremos de una colección de casos sobre los que razonar. Esta fase ha de ejecutarse en la etapa *update* introducida en 2.2.1.1 (véase Figura 2.4). Así, dispondremos de un *knowledge container* kc_{wcb} que emplearemos como base de casos.

4.3.1.1. Ejemplo

Imaginemos que el usuario solicita una receta de comida española que además contenga aceite de oliva, la típica dieta Mediterránea. Tendríamos $Q = \{cuisine = Spanishcuisine, ingredients = olive\ oil\}$. Suponiendo que ambos atributos están en Freebase (que lo están), dividiríamos el espacio de búsqueda en dos partes. Por un lado preguntaríamos por todos los platos de cocina española (B_1), y por otro solicitaríamos aquellos platos que contengan aceite de oliva como uno de sus ingredientes (B_2). De esta manera, obtenemos la diversidad suficiente para poder realizar una mejor adaptación de recetas. Una vez unidos los dos conjuntos, intentaremos rellenar el resto de atributos de cada plato, tarea no siempre posible como hemos explicado. Como resultado, obtendríamos 67 platos de comida española y 9 que empleen aceite de oliva, en total 76.

4.3.2. CombinedRetrieval

Una vez explicado el funcionamiento de la recuperación de casos de Freebase, continuamos con la integración de casos obtenidos de base local con los de base web. En el Capítulo 2, Secciones 2.2.1.2 y 2.2.2.1, se introdujo un modelo de base de casos mixta y se explicaron diversas maneras o políticas de cohesionar los casos. Una de ellas consistía en aprovechar la velocidad de ejecución de un sistema que trabaja en local, y si los resultados no eran lo suficientemente buenos, completarlos con información de la Web (cuya recuperación es, lógicamente, más lenta). Dejaba por implementar un algoritmo que denominamos *CombinedRetrieval*.

Tras la llegada de una consulta Q del usuario se procede a la recuperación de casos de la base local, en nuestro caso la base de ColibriCook. Obtendremos los k casos más similares a Q. Si las medidas de similitud de cada uno de los k casos es inferior a un cierto valor umbral prefijado, entonces procedemos a aplicar recuperación web, explicada en la Sección anterior. El conjunto de casos resultante será la unión de los recuperados de manera local con los descargados vía online. La elección de aplicar la unión, también puede cambiarse por otras como bien se explicó en la Sección 2.2.1.2.

Formalmente, el algoritmo $CombinedRetrieval(Q)$ se ejecuta en los siguientes pasos:

1. Sea $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ la query del usuario.

2. Sea $RS_{local} = Retrieve(Q, kc_{cb}, k)$ el resultado de recuperar k casos de la base de casos local.

3. si $similarity(RS(i)) < \delta, \forall i : 1 \leq i \leq k$ entonces

   a) $RS_{Web} = WebRetrieval(Q)$ (Algoritmo detallado en la Sección 4.3.1)

   b) $RS = RS_{local} \bigcup RS_{Web}$

   si no $RS = RS_{local}$

La utilidad de este algoritmo se aprecia con mayor claridad en un ejemplo como el siguiente.

4.3.2.1. Ejemplo

Imaginemos que deseamos comer *sushi* o algo similar. Crearemos una query $Q = \{cuisine = Japanese, ingredient = fish\}$. Si buscamos por comida oriental, ninguna de las recetas recuperadas es sushi porque no se encuentra en la base de casos local. Los resultados que nos proporiciona esta base son platos como *Chinese barbecue* u *Oriental Ginger chicken* que nada

tienen que ver con sushi. Si añadimos *fish* a la consulta obtenemos resultados como *Chinese Cashew Nut Prawns* o *Chinese Spiced Shrimp*, gambas y camarones respectivamente, como recetas más parecidas.

Por lo tanto, vamos a consultar si en Freebase hay algún plato más similar a lo que buscamos. Construimos una nueva query que se dividirá en dos partes como se explicó en la Sección anterior. Por un lado recuperará platos de comida japonesa y por el otro aquellos con pescado como uno de sus ingredientes. Los resultados son los siguientes: $Web_{results}$ ={**Sushi**, Fish and chips, Sashimi, Gefilte fish, Kedgeree, Nori, Tonkatsu, Soy sauce}. Tras realizar la unión de conjuntos con los recuperados de la base de casos local, se ejecutará la función de similitud k-NN (o cualquier otra disponible en jCOLIBRI) y se hará la pertinente adaptación. Queda demostrada la importancia de poder recurrir a otra nueva fuente de casos, que aunque sea más lenta, muchas veces nos dará el resultado que estamos buscando. Por último, la receta será almacenada o no en la base de casos local dependiendo de la política escogida para esta función.

4.3.3. WebAdaptation

En la Sección 2.2.2.2, advertíamos que no nos conformábamos con la utilización de información de la Red únicamente para la obtención de casos, es decir, rellenar el *knowledge container* kc_{cb} o kc_{wcb}. Pretendíamos también incorporar este conocimiento de alguna manera para poder emplearlo como conocimiento de adaptación. Un modo de llevar a cabo este propósito consiste en ayudarse de los casos de Freebase para la sustitución de atributos no deseados.

Proponemos aquí un algoritmo que se basa de conocimiento de Freebase para intentar obtener conocimiento de adaptación. Vamos a intentar adaptar un caso recuperado de la base local. Ahora utilizamos la fuente web para saber si podemos realizar una adaptación o no. Tras recuperar el caso más semajante de la base local e identificado el atributo a adaptar, procederemos a construir una *query* para Freebase. Esta *query* de Freebase se compondrá de los atributos que tienen en común la consulta de usuario y el caso recuperado más el atributo a sustituir. Si exisite una instancia en Freebase con la *query* de Freebase construída, significará que la sustitución planteada es correcta y se puede llevar a cabo.

Algoritmo **WebAdaptation(Q)**:

1. Sea $Q = \{\langle a_1, v_1\rangle, \langle a_2, v_2\rangle, \ldots, \langle a_n, v_n\rangle\}$ la query del usuario.

2. Sean $RS_{local} = Retrieve(Q, kc_{cb}, k)$ los resultados de recuperar k casos de la base de casos local.

3. $C = \{\langle a_1, v'_1\rangle, \ldots, \langle a_j, v'_j\rangle, \ldots, \langle a_n, v'_n\rangle\}$ es el 1-NN de RS_{local}.

4. Sea $\langle a_j, v'_j \rangle$ un atributo a sustituir en C, y v''_j el nuevo valor para a_j.

5. Sea $M = \{Atr(Q) \cap Atr(C) + \langle a_1, v''_1 \rangle\}$ el conjunto de valores de atributo en comun de Q y C más el atributo a sustituir.

6. Construyamos una query para Freebase Q_{FB} empleando los atributos en M.

7. Si existe un topic en Freebase para Q_{FB} significa que el atributo puede ser sustituido.

Para entender este algoritmo puede que sea necesario ayudarse de un ejemplo.

4.3.3.1. Ejemplo

Vamos a construir una query Q con el fin de optener algún plato que tenga *patata*. Vamos a asumir que estamos trabajando únicamente con la base de casos local. Más concretamente, nuestra query será $Q = \{ingredient_1 = potato, ingredient_2 = salt\}$. Ejecutando esta consulta sobre la base de casos de ColibriCook, obtenemos una solución como esta: $C = \{ingredients = \{chicken,\ raisin,\ macaroni,\ yam,\ scallion,\ mayonnaise,\ sour\ cream,\ red\ chilli,\ salt\}, type = salad\}$. Aquí, patata ha sido reemplazada por *yam* (batata). Nuestro algoritmo comprobará si la sustitución es correcta.

Por consiguiente, construimos una query para Freebase donde preguntaremos si existe otro plato de tipo *salad* con los ingredientes de la query Q que no están presentes en C más el candidato de la sustitución. De esta menera, buscaremos en Freebase si existe algún plato con $type\ of\ dish = salad$ e $ingredients = yam$. Existe uno, "Gado-gado". Ésto significa que *yam* se puede sustituir por *potato* porque hay otra ensalada con *yam* y *salt*.

El interés por el tema de la utilización de la Web para la adquisición de conocimiento de adaptación (AKA Adaptation Knowledge Acquisition) se está empezando a incrementar, y encontramos ya diversos estudios y propuestas. Por ejemplo, en Ihle et al. (2009), los autores intentan extraer conocimiento de adaptación para el mismo dominio que el que empleamos nosotros. Ellos se dirigen más al ámbito de páginas web específicas del dominio. En este tipo de sitios, los usuarios dejan sus experiencias sobre las recetas que hay propuestas y mediante técnicas de minería de datos intentan extraer los ingredientes de estas opiniones. Después, según cómo aparezcan estos ingredientes, disponen de unas reglas para intentar concluir si un ingrediente puede ser adaptado y cuál debería ser su sustituto, o también si hay un ingrediente a modificar. Digamos, que para el dominio de las recetas de cocina, esta idea es bastante buena, pero claro, es muy específica de este ámbito. Nosotros, con nuestro algoritmo, estamos intentando generalizar para poder utilizarlo en otros dominios posibles de los muchos que ofrece Freebase.

4.4. Integración en jCOLIBRI2

Tras ver los algoritmos que hemos propuesto para completar el modelo teórico del apartado 2.2, el siguiente paso consiste en realizar la integración sobre el framework jCOLIBRI2. Para explicar cómo se ha llevado a cabo esta operación, comenzaremos con una descripción del framework para conocer la parte de su funcionamiento y arquitectura más relevante para nuestro trabajo. Posteriormente, estudiaremos las modificaciones arquitectónicas y funcionales precisas para que esta plataforma pueda obtener la información que necesite de Freebase. Por último explicaremos, más detalladamente, cómo hemos construido el conector responsable del intercambio de información con la API de Freebase.

4.4.1. Introducción a jCOLIBRI

Comenzamos hablando sobre jCOLIBRI. Explicaremos brevemente su historia y evolución y luego pasaremos a una descripción de la versión actual, jCOLIBRI2, explicando los aspectos que es necesario conocer antes de pasar a la integración del conocimiento de Freebase.

4.4.1.1. Evolución

jCOLIBRI es la evolución de COLIBRI, un prototipo escrito en LISP, lo que limitaba su popularización. A partir de aquí, nació la primera versión de jCOLIBRI, ya escrita en Java. Incorporaba una interfaz gráfica, que ayudaba a los diseñadores de aplicaciones CBR. Estaba prevista para ser utilizada por diseñadores de sistemas de Razonamiento Basado en Casos sin conocimientos de programación [Recio-García et al. (2005)].

jCOLIBRI utiliza un lenguaje de programación basado en la terminología de CBROnto, una ontología que describe los principales términos de los sistemas CBR.

Mediante una sencilla GUI, el usuario es capaz de crear una estructura de datos que represente su sistema. Primeramente, se diseña la estructura de los casos, que constan de 3 partes: *descripción*, *solución* y *resultado*. Cada una de estas partes está compuesta por una serie de atributos. El usuario podrá elegir entre atributos simples y compuestos. Estos últimos no son más que composiciones de atributos simples. Para cada atributo simple se deberá proporcionar un nombre, tipo, peso y función de similitud, empleada en la comparación entre casos.

La persistencia de la base de casos se conseguía por diferentes vías, como ficheros de texto ASCII, archivos XML o bases de datos de MySql. Esta tarea correspondía a los conectores, que serán explicados posteriormente en mayor profundidad.

El diseño del sistema CBR se estructuraba en una secuencia de tareas

elegidas por el usuario. Se deben seleccionar cada una de las tareas que se deben ejecutar en cada una de las fases del ciclo de jCOLIBRI(pre-cycle, cycle, post-cycle) explicadas en Díaz-Agudo et al. (2007).

La segunda versión del framework, jCOLIBRI2, nace de la necesidad de de darle robustez a su arquitectura y además hacerla más sencilla de extender. Esta nueva versión no dispone de la opción de generación de aplicaciones desde interfaz gráfica. Los sistemas CBR diseñados con jCOLIBRI2 deben hacerse programando en lenguaje Java.

La última versión, la 2.1, incluye métodos para el desarrollo de sistemas recomendadores, además de corregir algunos problemas encontrados en la versión 2.0.

4.4.1.2. Descripción

jCOLIBRI2 es un framework escrito en Java que incorpora la posibilidad de construir sistemas CBR. Está preparado para ser ampliado con nuevas funcionalidades sin necesidad de modificar en exceso su arquitectura. La nueva versión diferencia claramente dos capas: una destinada a desarrolladores y otra para diseñadores, en donde se irán incluyendo nuevas funcionalidades.

En la primera capa, orientada a desarrolladores de aplicaciones CBR e incluye diferentes herramientas e interfaces gráficas para la creación y configuración de estos sistemas.

La segunda capa, la que más nos interesa, está dirigida a programadores y consta de las herramientas de gestión de conectores (para la persistencia de los datos), base de casos, núcleo de jCOLIBRI, librería de métodos y funciones de ayuda, ... La mayor parte de estas funcionalidades viene en modo de interfaces o clases abstractas que conforman el esqueleto del framework. Así, para la creación de una aplicación CBR desde el punto de vista del programador, habrá que seguir la estructura de clases preparada implementando los métodos necesarios de manera más concreta.

jCOLIBRI2 ejecuta un sistema CBR en tres pasos:

1. Precycle: Inicia la aplicación. Normalmente esta fase consta de la carga en memoria de la base de casos y la ejecución de algunos algoritmos de alto coste.

2. Cycle: Ejecuta el clásico ciclo de un sistema de Razonamiento Basado en Casos.

3. Postcycle: Se ocupa de la post-ejecución o el mantenimiento de código.

Estos pasos, vienen en forma de plantilla para ser implementados por los programadores.

4.4.2. Integración

En esta Sección procedemos a explicar de manera concreta las modificaciones necesarias en la plataforma jCOLIBRI2 y la construcción del conector para Freebase. Aunque existen otros frameworks de construcción de sistemas CBR como myCBR[Stahl y Roth-Berghofer (2008)] o IUCBRF[Bogaerts y Leake (2005)], jCOLIBRI es una referencia en la comunidad CBR, siendo utilizada en muchas universidades.

4.4.2.1. Modificaciones en la arquitectura

La arquitectura de jCOLIBRI2 divide el mantenimiento de los casos en dos capas: *la capa de persistencia* y *la capa de organización en memoria*. La primera de ellas gira en torno a los conectores. Los conectores se encargan de la gestión de información entre el medio físico de almacenamiento y la creación de los casos manejados por el sistema CBR que estemos construyendo. Los conectores que hay en jCOLIBRI pueden almacenar y recuperar información de diferentes medios como bases de datos SQL, archivos de texto, ontologías, ... La primera modificación es la creación de un nuevo conector. Ahora la gestión del medio físico no nos corresponde. Únicamente debemos crear un mecanismo de recuperar la información de una fuente Web y a partir de ella, extraer la información necesaria para la creación de un caso.

La segunda capa implicada, la de organización en memoria, define la estructura en que serán almacenados los casos para ser manipulados por la aplicación CBR. Disponemos de diversos esquemas de organización como organización lineal, lineal con caché, ... Cualquier tipo de organización en memoria puede ser empleado por cualquier tipo de conector. Por tanto, este aspecto no es necesario modificarlo. Únicamente, debemos centrarnos en la implementación del conector.

Como se explicó en las Secciones 2.2.1.2 y 2.2.2.1, manejamos la opción de distinguir entre dos *knowledge containers* distintos para la gestión de base de casos, uno para los casos locales kc_{cb}; y otro para almacenamiento de casos obtenidos vía online, denominado kc_{wcb}. Esto implica la división de la memoria en dos partes, una para contenedor. En principio, en los experimentos que hemos realizado, hemos fijado el mismo tipo de organización de las memorias para ambos contenedores. Esto no debería ser necesario, pues todos los tipos de organización siguen la misma interfaz.

4.4.2.2. Modificaciones funcionales

Una vez vistas las modificiaciones que hemos de realizar en la arquitectura de jCOLIBRI, pasamos a explicar qué modificaciones debemos aplicar al comportamiento del framework. En la Sección 4.4.1.2 explicamos los tres pasos que toda aplicación desarrollada sobre jCOLIBRI debe ejecutar: *pre-cycle*,

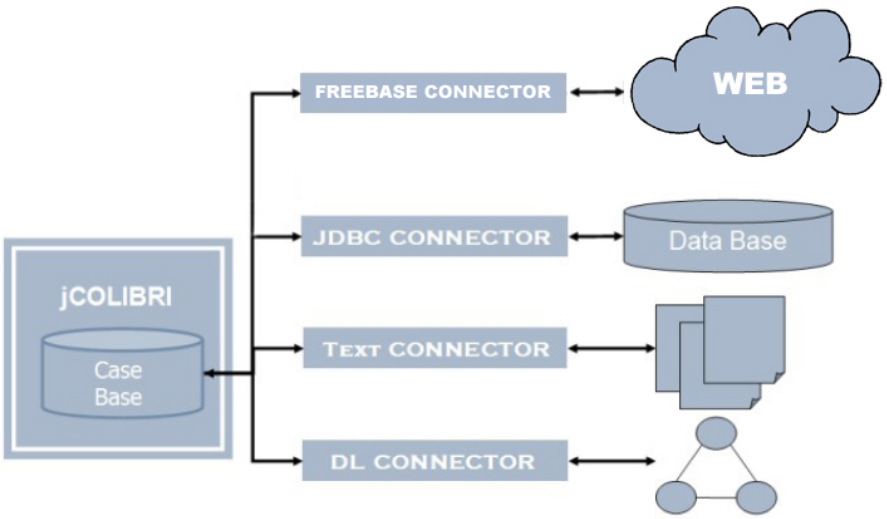

Figura 4.2: Integración del conector para Freebase.

*cycle*y *post-cycle*.

El principal cambio que se debe llevar a cabo implica lo referente al pre-ciclo. Esta fase se utiliza para la carga en memoria de los casos que tenemos almacenados en la base de casos. Cargar todos los casos de la fuente online, p.ej. Freebase, no tiene sentido. Ahora necesitamos cargar de memoria los casos referentes al dominio en el que estemos trabajando. Por tanto, es necesario crear una fase de actualización dependiente de la consulta del usuario. Por supuesto, el modelo híbrido que combina ambas bases de casos (Secciones 2.2.1.2 y 2.2.2.1) sigue siendo necesaria esta fase para la carga de la base de casos local.

Consecuentemente, la recuperación de casos online, que implica una actualización según la query del usuario, nos conduce a que este nuevo paso de *update* haya que incluirlo en la fase del *cycle* de jCOLIBRI. Realizamos una actualización de los *knowldedge containers* kc_{cb} o kc_{wcb} dependiendo si hemos escogido la separación o no de los mismos.

4.4.2.3. Conector para Freebase

Procedemos a explicar el diseño y funcionamiento del conector desarrollado para la recuperación de información de Freebase. Siguiendo la estructura de jCOLIBRI, hemos diseñado un conector que implementa la interfaz *Connector* para facilitar su integración y usabilidad.

El objetivo de esta herramienta es la creación de casos con información de Freebase (Figura 4.2). Un caso es un objeto CBRCase con cuatro atributos principales: Description, Solution, Result y Justification, todos ellos de tipo CaseComponent. El conector deberá rellenar el CaseComponent de la descripción de cada caso. Un CaseComponent no es más que un JavaBean con los atributos que tiene un caso. Lógicamente, el nombre de los atributos de un caso no tiene por qué coincidir con las propiedades de los *topics* de Freebase. Por consiguiente, lo primero que necesita el conector es conocer la correspondencia entre atributos del caso y nombre de las propiedades para preguntar a la fuente web. Siguiendo la estructura de los conectores de jCOLIBRI, realizaremos esta tarea mediante el uso de archivos XML, utilizando campos de modo:

```
<mapping>
<attribute>attribute1</attribute>
<fbprop>Freebase_property1</fbprop>

<attribute>attribute2</attribute>
<fbprop>Freebase_property2</fbprop>

<attribute>attribute3</attribute>
<fbprop>Freebase_property3</fbprop>
</mapping>
```

Así, lograremos un mapeo entre atributos y propiedades de Freebase. Una vez conozcamos el valor asociado a una propiedad de una instancia, identificar su atributo correspondiente en el caso será sencillo.

Con la intención de realizar el conector de la manera más genérica posible, el conector debería poder ser empleado para cualquier dominio de Freebase. Sin embargo, cada descripción de un caso tiene atributos diferentes y por tanto, el objeto CaseComponent que albergue dicha descripción será cada vez una clase Java distinta. Por tanto, hemos de indicar de alguna manera que clase Java corresponde a cada uno de los CaseComponent de un caso: description, solution, result y justification. Esto lo conseguimos también a través del archivo XML utilizado en la inicialización del conector. Se le deberá especificar las clases que debería instanciar en la creación de los casos. Ampliamos, por consiguiente, el archivo XML:

```
<type>Freebase_type_name</type>
<config>
   <descriptionclass>DescriptionClassName</descriptionClass>
   <descriptionmapping>
      <mapping>
        <attribute>attribute1</attribute>
```

```
        <fbprop>Freebase_property1</fbprop>

        <attribute>attribute2</attribute>
        <fbprop>Freebase_property2</fbprop>

          <attribute>attribute3</attribute>
          <fbprop>Freebase_property3</fbprop>
      </mapping>
   </descriptionmapping>

   <solutionclass>SolutionClassName</solutionclass>
   <solutionmapping>
      <mapping>
         <attribute>attribute</attribute>
         <fbprop>Freebase_property</fbprop>
         .
         .
         .
      </mapping>
   </solutionmapping>

   <resultclass>ResultClassName</resultclass>
   <resultmapping>...</resultmapping>

   <justificationclass>JustificationClassName</justificationclass>
   <justificationmapping>...</justificationmapping>
</config>
```

Una vez sabemos los nombres de las clases Java, por medio de técnicas de *Introspection*, podremos crear las instancias de los casos aunque no sepamos hasta que lleguemos a la ejecución el nombre de las clases. Destacamos también que en la anterior muestra de un posible archivo XML encontramos al comienzo la etiqueta <type>. La utilizamos para indicar en qué tipo de Freebase estamos trabajando, algo necesario a la hora de construir las queries para la fuente web.

El siguiente paso es la construcción de una query para Freebase, según la consulta que el usuario haya introducido. Un punto importante a tener en cuenta es que las consultas que hacemos a Freebase son consultas exactas, y no aproximadas como un sistema CBR. Por tanto si un usuario fija en una consulta 3 atributos (por ejemplo, en el dominio culinario: cocina española, ingrediente aceite y tipo de plato ensalada), los resultados devueltos por Freebase serán aquellas instancias que tienen esas tres propiedades con el valor correspondiente. Este resultado es demasiado pobre para poder realizar un razonamiento basado en casos y por esta razon, hemos implementado el

algoritmo *WebRetrieval* de la Sección 4.3.1.

Por tanto, nos disponemos a realizar una partición del espacio de búsqueda. Dividiremos la query por cada atributo fijado por el usuario y recopilaremos los resultados de preguntar únicamente por una propiedad. A continuación, realizaremos la unión de conjuntos. Así, tendremos un abanico de casos más amplio sobre el que razonar.

Cada consulta que hagamos a Freebase, deberá realizarse por medio de los Servicios Web ofrecidos por la página. Esto equivale a la construcción de una query en lenguaje MQL (Sección 3.3) que será enviada vía HTTP, de modo similar a lo que realiza un explorador Web (de hecho, se puede probar una query de Freebase en cualquier navegador y funcionará perfectamente[2]). Freebase nos devolverá un objeto JSON[3]. Un objeto JSON es un conjunto desordenado de pares nombre/valor. Un objeto comienza con (llave de apertura) y termine con (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma). En ese objeto JSON vendrá tanto la información que hemos proporcionado (los atributos que hemos fijado) como los datos que estamos buscando. El último paso, por tanto, consisitirá en la extracción de los valores de los campos que dejamos indefinidos al construir la query de Freebase. Seguidamente, estructuraremos toda la información proporcionada en forma de casos como se ha indicado al comienzo de esta Subsección. Concluida la construcción de todos los casos, dispondremos de una nueva base de casos con la que poder trabajar.

4.5. Conclusiones

En este Capítulo hemos explicado todo lo referente al Marco Práctico de nuestro proyecto. Hemos comenzado por una descripción del dominio de trabajo en el que implementaremos los algoritmos, estudiando la base de casos local y el dominio de Freebase al que recurriremos. Posteriormente, se han propuesto tres algoritmos para la explotación de la información que alberga Freebase. Estos mismos algoritmos son los que quedaron pendientes de desarrollar cuando explicábamos el modelo teórico que queríamos seguir para el WEB-CBR (Sección 2.2). Hemos concluido el Capítulo exponiendo la integración en jCOLIBRI2, destacando el proceso de creación del conector para Freebase. Con esto, damos por finalizado el grueso del proceso práctico del trabajo. A continuación, pasaremos a realizar una serie de experimentos para comprobar su funcionamiento y utilidad.

[2] `http://www.freebase.com/api/service/mqlread?query={"query":{"type":"/music/artist","name":"ThePolice","album":[]}}`

[3] http://json.org

Capítulo 5

Validación Experimental

Resumen: En este Capítulo se plantearán dos hipótesis al comienzo y se realizarán dos experimientos intentando demostrar ambas.

5.1. Hipótesis

Nos disponemos a comprobar el funcionamiento del Modelo Práctico detallado en el Capítulo 4. Para ello vamos a formular dos hipótesis, a las que intentaremos dar respuesta con la implementación de nuestro proyecto trabajando de nuevo en el dominio de las recetas de cocina (Sección 4.2).

Para el primer experimento volveremos a los casos estudiados en las Secciones 2.2.1.1 y 2.2.1.2, que proponían emplear el conocimiento online en la recuperación de casos y trabajar con los modelos de base de casos única y base de casos mixta, respectivamente. Realizaremos, por tanto, una evaluación por separado de las bases de casos local y web, y seguidamente, el mismo estudio sobre el modelo conjunto.

Lo que pretendemos comprobar en este primer experimento será *si la precisión de los casos de la base de casos local se ve mejorada con la inclusión de casos obtenidos de la Web.*

Tras este expermiento, y a tenor de los resultados obtenidos en el mismo, nos plantearemos una segunda hipótesis. Querremos comprobar *si el recall de los casos de la base de casos local se ve mejorado con la inclusión de casos obtenidos de la Web.*

5.2. Primer experimento

Este primer experimento intentará responder a la hipótesis planteada en la Sección anterior. Necesitaremos una base de casos local, la que utiliza ColibriCook, y el conector de Freebase trabajando dentro del dominio Food

& Drink. Nos disponemos a comprobar si añadiendo una base de casos online mejoramos la precisión de los casos recuperados. Evaluaremos primero las dos bases de casos por separado y, posteriormente, las uniremos para comprobar su eficacia.

Las evaluaciones consisten en un problema de clasificación. Se extraeran de la base de casos uno o varios casos para emplearlos como queries. Cada query constará de la lista de ingredientes de una receta, y se buscará obtener el tipo de cocina de esa misma receta. Para ello, después de aplicar recuperación K-NN, se votará por mayoría que clase se espera conseguir. Dependiendo si se acierta o no, se calcularán las tasas de acierto correspondientes. Los métodos de evaluación están disponibles en el framework de jCOLIBRI2, y son los siguientes:

- *Leave One Out*: Este método extrae un caso C_i de la base de casos y ese mismo caso lo convierte en una query omitiendo uno de sus atributos j. Queda una query $Q = \{C_i(1), C_i(2), \ldots, C_i(j-1), C_i(j+1), \ldots, C_i(n)\}$ Ejecutando la función de similitud que le indiquemos al resto de los atributos, nos devolverá en caso más parecido que resta en la base de casos C_k. Si $C_i(j) = C_k(j)$ sumaremos un acierto. En caso contrario, sumaremos un fallo. Repetiremos el proceso para todos los casos de la base de casos.

- *N-Fold*: Consiste en dividir la base de casos en bloques o *folds* de tamaño N. Cada uno de los bloques será considerado como conjunto de queries y se intentará evaluar sobre el resto de la base de casos de igual modo que el método anterior, anotando igualmente la tasa de aciertos y de fallos.

Nuestros experimentos manejarán los atributos de ingredientes y tipo de cocina. El caso que se evaluará como query lo dividiremos en dos partes. La query en sí serán los ingredientes que tiene el caso y el tipo de cocina será lo que queremos que nos devuelva el método de evaluación que ejecutemos. Por supuesto, hemos tenido que eliminar aquellas recetas que no estuvieran completas, es decir, que tuvieran alguno de sus atributos vacíos. A cada atributo que se utilice como parte de la query ha de asignársele una función de similitud. En nuestro caso, para un conjunto de ingredientes hemos utilizado la siguiente fórmula:

Para dos conjuntos de ingredientes A y B, la función de similitud entre ambos vendrá dada por la fórmula siguiente: $|A \bigcap B| \,/\, |A \bigcup B|$

Primeramente, comenzamos evaluando la base de casos local. Para el método *Leave One Out* obtenemos una precisión del 79.96 % (véase Figura 5.1), mientras que para el método N-Fold[1] conseguimos 79.29 %.

[1] Todos los resultados aquí expuestos de N-Fold tienen $N = 10$

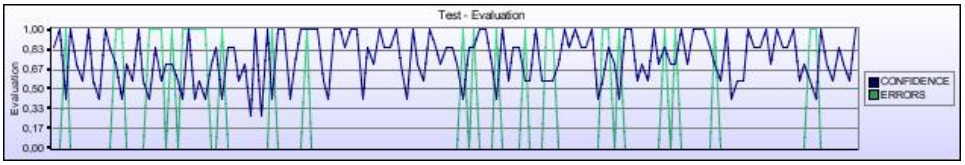

Figura 5.1: Resultados para Leave One Out sobre base local.

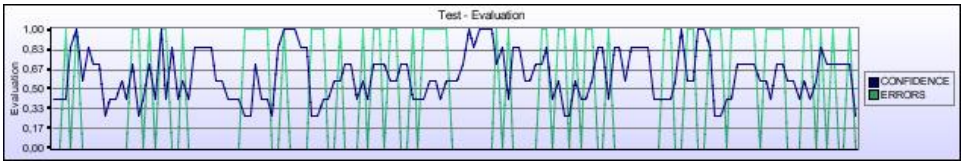

Figura 5.2: Resultados para Leave One Out sobre base web.

Las primeras evaluaciones de la base de casos web fueron muy malas y que al juntar las dos bases de casos iban a ser aún peores. Optamos por hacer una normalización del tipo de cocina de cada uno de los platos de Freebase. Con esto nos ajustábamos a los tipos de cocina de ColibriCook (Normalización en el Apéndice A). Gracias a esto, reducimos el número de tipos de cocina de 47 a 5: *Indian*, *American*, *Oriental*, *Mediterranean* y *Other*.

Con esta normalización, obtuvimos unos resultados en precisión de 58.33 % con el método Leave One Out (véase Figura 5.2) y 45.49 % con N-Fold. Unos resultados bastante pobres. Por último, si mezclamos ambas bases de casos para comprobar si se mejora la precisión, que es lo que en un principio nos preguntábamos, conseguimos 69.44 % (véase Figura 5.3)y 62.29 % para Leave One Out y N-Fold respectivamente. En cada gráfica están indicados el error y la confianza (*confidence*). El valor del error únicamente puede tomar valores de 0 ó 1. A lo largo del eje X se distribuyen los intentos de clasificación. Un valor de error de 0 significará que la clasificación de la query evaluada ha sido correcta y viceversa. El valor de confianza indica el porcentaje de casos que predecían la clase del elemento que intentaban clasificar.

Este primer experimento no corroboró la hipótesis que nos planteábamos en la Sección 5.1. Agregando una base de casos online no sólo no mejoramos la precisión, sino que la empeoramos, y eso que hicimos una normalización de atributos para intentar obtener mejores resultados. No comprendíamos cómo podíamos haber conseguido resultados tan pobres y decidimos inspeccionar a fondo las posibles causas. Éstas son las posibles explicaciones:

- Muchas de las recetas de Freebase tienen uno o dos ingredientes únicamente, algunos tan generales como *pork* (cerdo) o *bread* (pan). Estos mismos ingredientes se encuentran con facilidad en otros tipos de cocina y claro, intentar predecir el tipo de cocina solamente con este

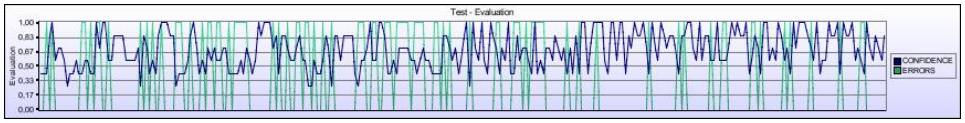

Figura 5.3: Resultados para Leave One Out sobre base mixta.

ingrediente se antoja muy complicado.

- El problema de los derivados: En Freebase encontramos, por ejemplo, varios tipos de arroz dentro de la comida oriental. El arroz es un ingrediente que identifica claramente con este tipo de cocina, pero al realizar evaluaciones no se consideran como el mismo ingrediente. Este problema también se encuentra con otros ingredientes como huevo, huevo cocido, cebolla, cebolla frita, ... Este problema se podría solucionar mediante el uso de algún tipo de ontología.

- Hay unas pocas recetas referentes a bebidas que no se asemejan a ninguna otra. Es complicado inferir su procedencia.

- Entre las cocinas mediterráneas y orientales se repiten muchos ingredientes tales como cebolla, huevo o tomate. Luego, cada una de ellas tiene su propio ingrediente principal que las hace diferentes y que caracteriza su tipo de cocina. Lamentablemente, este ingrediente no se vuelve a repetir en ninguna otra receta, lo cual no hace aumentar el valor de similitud con recetas de su mismo tipo de cocina.

- En el tipo de cocina INDIAN sólo se repiten estos ingredientes: *rice*, *lentil*, *atta flour*, *coconut milk* y *chicken*. Además suelen aparecer como únicos ingredientes de algunos platos. Así, es complicado caracterizar este tipo de cocina.

Esto nos lleva a pensar si era tan descabellada nuestra hipótesis de que con la base web mejoraríamos el rendimiento de nuestro sistema, y más viendo los resultados obtenidos por Leake y Sooriamurthi (2003), que en sus estudios con sistemas Multi-CBR conseguían el resultado opuesto. Pero claro, ellos en su estudio, utilizan diversas bases de casos que probablemente se parezcan ambas a la que nosotros empleamos como local, la de ColibriCook. Los resultados en solitario de nuestra base de casos local también son buenos. Si observamos las recetas que maneja ColibriCook, nos damos cuenta de que son muy *completas*: cada receta posee una buena lista de ingredientes, de los cuales no hay problema de derivados y para el mismo tipo de cocina se repiten bastantes ingredientes. Podríamos decir que el espacio de búsqueda de la base de casos local está mucho más concentrado y es más completo, mientras que el espacio de búsqueda de la base Web es mucho más amplio, pero también está más disperso.

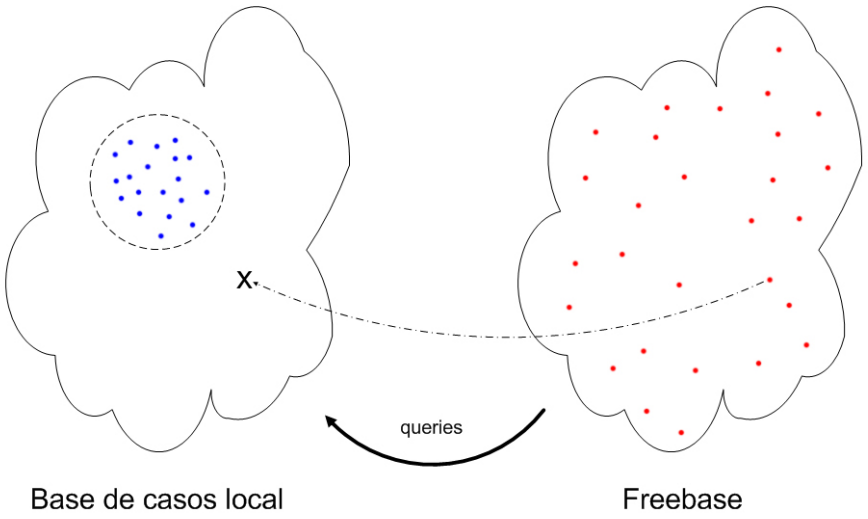

Figura 5.4: Modelo a seguir en el segundo experimento.

5.3. Segundo Experimento

A tenor de los resultados obtenidos en el experimento anterior, concluimos que la base de casos local era más específica y la base online mucho más variada y dispersa a la vez. Vamos a intentar demostrar esto. Vamos a intentar comprobar que la tasa de acierto será mayor al intentar evaluar casos de la fuente web sobre la fuente local que al revés. Esto demostraría que aunque la fuente local es muy completa, también es menos diversa. Por lo tanto, la complementación de la base de casos local con casos web mejorará el *recall* del sistema CBR.

Utilizaremos unas medidas de *recall* similares a las propuestas por McSherry (2001), que lo define como:

$$recall(S, L) = \frac{\sum_{C \in L} r(C, S, L)}{|L|} \cdot 100$$

$$\text{donde } r(C, S, L) = \begin{cases} 1 & \text{si } C \in rCases(C^*, S, L) \\ 0 & \text{en otro caso} \end{cases}$$

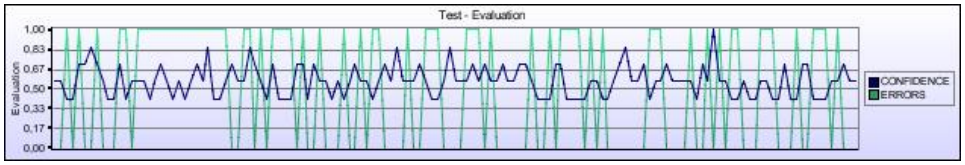

Figura 5.5: Evaluación de queries de ColibriCook sobre Freebase. Cocinas normalizadas. Mismo número de recetas.

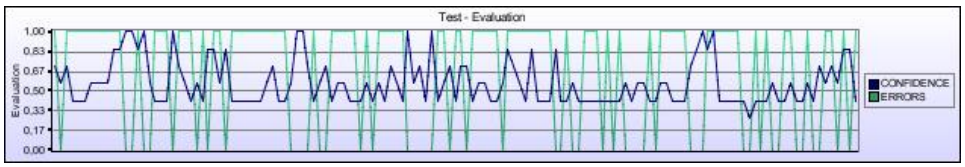

Figura 5.6: Evaluación de queries de Freebase sobre ColibriCook. Cocinas normalizadas. Mismo número de recetas.

siendo $rCases(C_t, S, L)$ el conjunto de casos recuperados cuando se aplica la estrategia S al conjunto L, teniendo como objetivo el caso C_t.

Para ello, vamos a hacer una modificación del método de evaluación *Leave One Out*, estrategia (S). Extraeremos uno a uno todos los casos de la base de casos local para utilizarlos como query, pero esta vez los evaluaremos sobre la base de casos de Freebase. Posteriormente, lo haremos en sentido inverso como se aprecia en la Figura 5.4.

Primero, mantendremos las cocinas originales de Freebase, es decir, no realizaremos la normalización explicada en el apartado anterior. Lo que si que haremos aquí será coger el mismo número de recetas de ambas bases de casos. Evaluando los casos de ColibriCook sobre Freebase, obtenemos una tasa de 47.2 %, mientras que si cogemos casos de Freebase y los intentamos evaluar sobre la base local únicamente llegamos al 24 %.

El siguiente paso es simplificar los tipos de cocina, es decir, aplicar la normalización, pero esta vez evaluaremos todas las recetas que podemos encontrar en ambas bases de casos. Con queries obtenidas de ColibriCook y evaluadas sobre Freebase logramos alcanzar un 40.2 %, mientras que en sentido inverso, 23 %.

Por último, vamos a realizar el experimento con los datos más semejantes todavía. Aplicamos la normalización para los tipos de cocina de Freebase y además seleccionaremos el mismo número de casos de ambas bases. Así los porcentajes aumentan a 51.8 % (Figura 5.5) y 36.5 % (Figura 5.6), utilizando como consultas casos extraídos de las bases local y online respectivamente.

La Tabla 5.1 y la Figura 5.7 resumen las tasas de error alcanzadas.

Resultados del experimento		
Cocinas y número de recetas	Queries de	Tasa de acierto
Cocinas originales y mismo número de recetas	Local Web	47.2 % 24.3 %
Cocinas normalizadas y distinto número de recetas	Local Web	40.2 % 23.0 %
Cocinas normalizadas y mismo número de recetas	Local Web	51.8 % 36.5 %

Tabla 5.1: Resultados de *recall* para el segundo experimento.

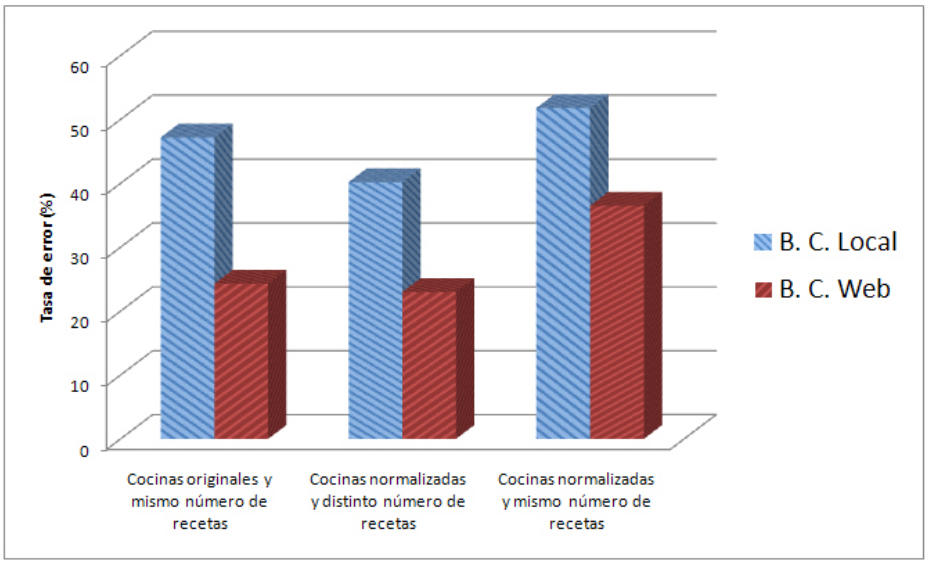

Figura 5.7: Resultados del segundo experimento.

Con este experimento, hemos demostrado que al utilizar queries más generales sobre una base de casos con conocimiento más específico, aunque más completo, obtendremos peores resultados que evaluándolos sobre una fuente más amplia aunque más dispersa. Por lo tanto, la inclusión de casos Web mejora significativamente el *recall* del sistema CBR. La situación queda reflejada en la figura 5.4.

5.4. Conclusiones

En este Capítulo de Experimentos hemos planteado dos hipótesis al comienzo y hemos intentado demostrarlas. La primera no ha sido posible, y hemos llegado a la conclusión de que el conocimiento de las bases locales es más completo y está más concentrado que el de las fuentes online. Tras esto, hemos intentado comprobar qué ocurriría con el *recall*. Los resultados nos

indican que al evaluar los casos de una base local en una base web, se obtienen peores resultados que al revés. Esto es debido a que el conocimiento web es menos preciso pero más extenso y disperso que las bases de casos locales.

Capítulo 6

Conclusiones y trabajo futuro

Resumen: En este último Capítulo resumiremos las ideas principales seguidas en el proyecto indicando las concluiones alcanzadas para cada una de ellas. Seguidamente, indicaremos unas líneas de trabajo futuro señalando los próximos pasos a seguir para este trabajo.

6.1. Conclusiones

Como señalábamos al comienzo de este trabajo, mucha de la información que hay almacenada en la web pertenece a experiencias y opiniones de usuarios, sobre todo a raíz de la llegada de la *Web 2.0*. Ser capaces de recopilar esta información sería de gran ayuda a la hora de la construcción de bases de casos, una tarea complicada dentro del mundo del Razonamiento Basado en Casos.

En este proyecto se ha expuesto de manera teórica las modificaciones estructurales y funcionales necesarias para la inclusión de conocimiento web en el framework de jCOLIBRI2. Se han estudiado diferentes fuentes web, intentando seleccionar aquella que ofreciera una integración viable en el framework, pero sobre todo, buscando que fuera representativa como una fuente de casos. Hemos evitado seleccionar fuentes de conocimiento específico sobre un dominio concreto, y así, nuestra elección Freebase, aunque ha sido testada para un único dominio, ofrece posibilidades de utilizar el conector implementado para otros campos.

Se comenzó con una introducción al Razonamiento Basado en Casos (Capítulo 2), haciendo hincapié en el ciclo CBR y en la organización del conocimiento general de los sistemas CBR, al que organizábamos en diferentes *knowledge containers*. También se realizaron diversas descripciones de la organización de las bases de casos, de sistemas CBR multiagente y de aportaciones previas al WEB-CBR. A continuación, se pasó a explicar el modelo

teórico propuesto para WEB-CBR. Comenzamos por los aspectos estructurales, estudiando cómo emplear el conocimiento Web en forma de base de casos. Se introdujeron dos modelos de funcionamiento: emplear la Web como fuente de casos exclusiva para el sistema o combinar la base de casos Web con una base de casos local, dando así el un model mixto. Seguidamente, se habló de los aspectos funcionales que se verían afectados en este modelo. Se propuso una estrategia de combinar casos locales con casos online para intentar mejorar tiempos de ejecución y así recurrir sólo a la fuente Web cuando fuera preciso. También se propuso la utilización de la Web como una fuente de conocimiento de Adaptación. Se plantearon tres algoritmos: *WebRetrieval*, *CombinedRetrieval* y *WebAdaptation* en espera de que fuesen implementados en el Marco Práctico del trabajo. Por último, se comentaron aspectos a tener en cuenta acerca de la gestión y el mantenimiento de la base de casos al incluir una fuente Web en un sistema CBR.

Tras el estudio de diferentes fuentes web que proporcionaban información estructurada, y después de decantarnos por una de ellas (Capítulo 3), se pasó al desarrollo del Marco Práctico del proyecto. Se implementaron los tres algoritmos que se proponían en el Capítulo 2. Comenzando con el de *WebRetrieval*, se desarrollo la manera de recuperar *topics* de Freebase, que serían los casos del sistema CBR, a partir de una query del usuario. Luego, desarrollamos un algoritmo *CombinedRetrieval* que trabajaba con recuperación de casos de la base local y si ninguno de ellos superaba un valor umbral de similitud con la consulta, entonces se recurría al algoritmo anterior de WebRetrieval. Finalmente, se explicó un algoritmo para aprovechar la información de Freebase en forma de conocimiento de adaptación. Se desarrolló un método para la validación de adaptaciones.

Se han encontrado problemas, principalmente señalando los tiempos de carga y la *calidad* de la información encontrada. Estos problemas los encontrará todo aquel que trate de enfrentarse a obtención de información de la Web. Aún así, hemos intentado minimizar estos problemas en la medida posible, introduciendo modelos mixtos de bases de casos para no exceder los tiempos de carga; y separando información local de online en diferentes *knowledge containers* evitando así la inclusión de excesivo ruido en la base local. Los algoritmos propuestos en el Capítulo 4, han sido probados para el dominio Food & Drink de Freebase, quedando como trabajo futuro la realización de más experimentos sobre otros dominios.

En el Capítulo 5, se desarrollaron dos experimentos. El objetivo de estos experimentos era comprobar si las medidas de precisión y de *recall* mejoraban tras la integración de una base de casos online. Para ello, se realizaron problemas de clasificación y se emplearon los métodos de *Leave One Out* y de *N-Fold*. En el primer experimento se intentó comprobar una posible mejora de la precisión. Para ellos se evaluaron primero las bases local y Web por separado, y luego, el modelo mixto. Para el segundo experimento que-

ríamos comprobar si el *recall* se vería afectado. Construimos una variación del método *Leave One Out* en el que los casos de una base de casos pasaban a evaluarse como queries en la otra.

Hemos demostrado que el espacio de búsqueda ocupado por casos extraídos de Freebase es mucho más amplio que el que puede ofrecer una base de casos local. No obstante, la concentración de los casos locales es una situación más complicada de encontrar en fuentes de información web. Aún así, una buena combinación y utilización de ambas (modelo mixto de la Sección 2.2.1.2 se antoja la solución más interesante. Cuando la concentración de casos de una base local no guíe a una buena solución, la opción de tomar referencias de un abanico más amplio puede ser una buena estrategia. Por supuesto, conviene realizar estudios como los experimentos del Capítulo 5 para otros dominios de Freebase y sacar las convenientes conclusiones.

Presentamos de manera esquematizada las aportaciones más importantes de este proyecto:

1. Estudio de aproximaciones previas al WEB-CBR.
2. Planteamiento de modificaciones funcionales que supondría la integración de conocimiento Web en el CBR.
3. Planteamiento de modificaciones estructurales que supondría la integración de conocimiento Web en el CBR.
4. Estudio de fuentes Web.
5. Creación de un conector de Freebase para jCOLIBRI.
6. Planteamiento de diversos algoritmos para WEB-CBR.
7. Validación experimental de dichos algoritmos.

6.2. Trabajo futuro

Aquí explicaremos unas líneas de trabajo futuro para poder continuar el proyecto:

- Como acabamos de explicar en la Sección anterior, un buen punto para continuar el trabajo sería la de realizar experimentos como los tratados en el Capítulo 5 pero incluyendo otros dominios de Freebase.
- Otra buena opción sería la de diseñar más técnicas de explotación del conocimiento de Freebase, centrándonos en algoritmos de adaptación.
- Una buena idea para mejorar los resultados obtenidos en los experimentos, sería incluir una ontología para trabajar con los ingredientes,

evitando así problemas de derivados, e incrementando a las tasas de acierto obtenidas en los problemas de clasificación.

- Mucha de la información referente a experiencias de usuarios se encuentra en páginas colaborativas como blogs. Poder recopilar información de blogs sería una opción muy interesante y una buena fuente de conocimiento de adaptación. Sin embargo, de momento sólo podríamos aplicar técnicas de Procesamiento del Lenguaje Natural y Extracción de Información, tarea considerablemente difícil.

- Explorar más fuentes de información estructuradas que vayan saliendo a la luz y esperar a la publicación de una API para la plataforma de Google Squared, que fue la principal razón por la que fue descartada en la Sección 3.3.

- No hemos utilizado el conocimiento adquirido para potenciar el *knowledge container* kc_{ret}, que compete a las funciones de similitud empleadas en las tareas de recuperación. Buscar algún modo de mejorar este conocimiento sería una opción muy interesante.

En conclusión, este trabajo intenta aportar un marco en el que organizar las distintas posibilidades de la investigación en el prometedor campo del WEB-CBR. Para ello, se estudian, analizan y clasifican sus características más importantes. Además se proponen y contrastan una serie de algoritmos que han sido integrados dentro de la plataforma JCOLIBRI para su divulgación en la comunidad CBR.

Apéndice A

Normalización de tipos de cocina

Resumen: Aquí incluimos el listado de tipos de cocina de Freebase que utilizamos en los experimentos del Capítulo 5 y cómo los hemos normalizado para alguno de los apartados de dicho Capítulo.

Cocina de Freebase	**Cocina normalizada**
Italian	Mediterranean
Chinese American	Oriental
Jewish Cuisine	Mediterranean
Singaporean Cuisine	Oriental
American	American
British Cuisine	Other
French	Other
Indonesian	Oriental
Malaysian	Oriental
Indian	Indian
Irish	Other
Mexican	American
Thai	Oriental
Japanese Cuisine	Oriental
Korean	Oriental
Greek	Mediterranean
Filipino	Oriental
Cajun	American
Cuban	American
Chinese	Oriental
Cuisine of Hong Kong	Oriental

Armenian	Other
Nepal	Indian
German	Other
New Zealand Cuisine	Other
Tamil	Indian
Jewish	Mediterranean
Ukraine	Other
Arab Cuisine	Mediterranean
Polish	Other
Swiss	Other
Roman	Mediterranean
South Indian Cuisine	Other
Vietnamese	Oriental
Goan Cuisine	Indian
Scottish Cuisine	Other
Welsh Cuisine	Other
Belgian	Other
Tiki Bar	Oriental
Spanish Cuisine	Mediterranean
Midwestern United States	American
Hong Kong	Oriental
Cuisine of the Midwestern United States	American
Taiwanese	Oriental
Cuisine of New England	American
Apulian	Mediterranean
Southern United States	American
Oriental	Oriental
Mediterranean	Mediterranean

Tabla A.1: Normalizaciones del tipo de cocina

Bibliografía

Aamodt, A. y Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun.*, vol. 7(1), páginas 39–59, 1994.

Althoff, K., Auriol, E. y Manago, M. A Review of Industrial Case-Based Reasoning Tools. *AI Intelligence*, 1995.

d' Aquin, M., Badra, F., Lafrogne, S., Lieber, J., Napoli, A. y Szathmary, L. Case base mining for adaptation knowledge acquisition. En *IJCAI* (editado por M. M. Veloso), páginas 750–755. 2007.

Asiimwe, S., Craw, S., Taylor, B. y Wiratunga, N. Case authoring: From textual reports to knowledge-rich cases. En *ICCBR*, páginas 179–193. 2007.

Berners-Lee, T., Hendler, J. y Lassila, O. The Semantic Web. *Scientific American*, vol. 284(5), páginas 34–43, 2001.

Bogaerts, S. y Leake, D. B. Increasing ai project effectiveness with reusable code frameworks: A case study using iucbrf. En *FLAIRS Conference*, páginas 2–7. 2005.

Díaz-Agudo, B. *Una aproximación ontológica al desarrollo de sistemas de razonamiento basado en casos*. Phd, Universidad Complutense de Madrid, 2002.

DeMiguel, J., Plaza, L. y Díaz-Agudo, B. Colibricook: A cbr system for ontology-based recipe retrieval and adaptation. En *ECCBR Workshops*, páginas 199–208. 2008.

Díaz-Agudo, B., González-Calero, P. A., Recio-García, J. A. y Sánchez-Ruiz-Granados, A. A. Building cbr systems with jcolibri. *Sci. Comput. Program.*, vol. 69(1-3), páginas 68–75, 2007.

Hanney, K. y Keane, M. The Adaptation Knowledge Bottleneck: How to Ease it by Learning from Cases. En *Proceedings of the Second International Conference on Case-Based Reasoning.*. Springer Verlag, 1997.

HERRERA, P. J., IGLESIAS, P., ROMERO, D., RUBIO, I. y DÍAZ-AGUDO, B. Jadacook: Java application developed and cooked over ontological knowledge. En *ECCBR Workshops* (editado por M. Schaaf), páginas 209–218. 2008.

IHLE, N., HANFT, A. y ALTHOFF, K.-D. Extraction of adaptation knowledge from internet communities. En *ICCBR 2009 Workshop Proc., Workshop Reasoning from Experiences on the Web* (editado por S. J. Delany), página to appear. 2009.

KOLODNER, J. L. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, vol. 7, páginas 243–280, 1983.

LEAKE, D. B. y POWELL, J. H. Mining large-scale knowledge sources for case adaptation knowledge. En *ICCBR*, páginas 209–223. 2007.

LEAKE, D. B. y POWELL, J. H. Knowledge planning and learned personalization for web-based case adaptation. En *ECCBR*, páginas 284–298. 2008.

LEAKE, D. B. y SOORIAMURTHI, R. When two case bases are better than one: Exploiting multiple case bases. En *ICCBR* (editado por D. W. Aha y I. Watson), vol. 2080 de *Lecture Notes in Computer Science*, páginas 321–335. Springer, 2001. ISBN 3-540-42358-3.

LEAKE, D. B. y SOORIAMURTHI, R. Automatically selecting strategies for multi-case-base reasoning. En *ECCBR*, páginas 204–233. 2002.

LEAKE, D. B. y SOORIAMURTHI, R. Dispatching cases versus merging case-bases: When mcbr matters. En *FLAIRS Conference*, páginas 129–133. 2003.

DE MANTARAS, R. L., MCSHERRY, D., BRIDGE, D., LEAKE, D., SMYTH, B., CRAW, S., FALTINGS, B., MAHER, M. L., COX, M. T., FORBUS, K., KEANE, M., AAMODT, A. y WATSON, I. Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review*, vol. 20(3), páginas 215–240, 2006.

MCSHERRY, D. Demand-Driven Discovery of Adaptation Knowledge. En *Proceedings of the 16 International Joint Conference on Artificial Intelligence*, páginas 222–227. 1999.

MCSHERRY, D. Precision and recall in interactive case-based reasoning. En *ICCBR*, páginas 392–406. 2001.

PLAZA, E. Semantics and experience in the future web. En *ECCBR*, páginas 44–58. 2008.

PLAZA, E. y BACCIGALUPO, C. Principle and praxis in the experience web: A case study in social music. 2009.

PLAZA, E. y MCGINTY, L. Distributed case-based reasoning. *Knowledge Eng. Review*, vol. 20(3), páginas 261–265, 2005.

PRASAD, M. V. N. Distributed case-based learning. En *ICMAS*, páginas 222–230. IEEE Computer Society, 2000. ISBN 0-7695-0625-9.

RECIO-GARCÍA, J. A., CASADO-HERNÁNDEZ, M. A. y DÍAZ-AGUDO, B. Extending cbr with multiple knowledge sources from web. 2010.

RECIO-GARCÍA, J. A., SÁNCHEZ-RUIZ, A. A., DÍAZ-AGUDO, B. y GONZÁLEZ-CALERO, P. A. jcolibri 1.0 in a nutshell. a software tool for designing cbr systems. En *Proccedings of the 10th UK Workshop on Case Based Reasoning* (editado por M. Petridis), páginas 20–28. CMS Press,University of Greenwich, 2005. ISBN 1-904521-30-4.

RICHTER, M. M. The knowledge contained in similarity measures. Invited Talk at the First International Conference on Case-Based Reasoning, ICCBR'95, Sesimbra, Portugal, 1995. `http://wwwagr.informatik.uni-kl.de/~?lsa/CBR/Richtericcbr95remarks.html` [Last access: 2002-10-18].

SCIME, A. *Web mining : applications and techniques / Anthony Scime [editor]*. Hershey, Pa. ; London : Idea, 2005. ISBN 1591404142. Formerly CIP.

SHADBOLT, N., BERNERS-LEE, T. y HALL, W. The semantic web revisited. *IEEE Intelligent Systems*, vol. 21(3), páginas 96–101, 2006.

SMYTH, B., CHAMPIN, P.-A., BRIGGS, P. y COYLE, M. The case-based experience web. En *In: the WebCBR-09 Reasoning from Experiences on the Web Workshop at the 8th International Conference on Case-Based Reasoning (ICCBR 2009)*. Seatle, Washington, USA, 2009.

STAHL, A. y ROTH-BERGHOFER, T. Rapid prototyping of cbr applications with the open source tool mycbr. En *ECCBR* (editado por K.-D. Althoff, R. Bergmann, M. Minor y A. Hanft), vol. 5239 de *Lecture Notes in Computer Science*, páginas 615–629. Springer, 2008. ISBN 978-3-540-85501-9.

WILKE, W., VOLLRATH, I., DIETER ALTHOFF, K. y BERGMANN, R. A framework for learning adaptation knowledge based on knowledge light approaches. En *In 5th German Workshop on CBR*, páginas 235–242. 1996.

WIRATUNGA, N., KOYCHEV, I. y MASSIE, S. Feature selection and generalisation for retrieval of textual cases. En *ECCBR*, páginas 806–820. 2004.

WIRATUNGA, N., LOTHIAN, R. y MASSIE, S. Unsupervised feature selection for text data. En *ECCBR*, páginas 340–354. 2006.

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: «Integración de Conocimiento Web en Sistemas de Razonamiento Basado en Casos», realizado durante el curso académico 2009-2010 bajo la dirección de Belén Díaz Agudo y con la colaboración externa de dirección de Juan Antonio Recio García en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Miguel Ángel Casado Hernández

Madrid, 21 de Junio de 2010