

Actas del Taller de Trabajo Zoco'07/JISBD
Integración de Aplicaciones Web no Desmantelables

XII Jornadas de Ingeniería del Software y Bases de Datos
Zaragoza, 11 de setiembre de 2007



<http://www.tdg-seville.info/CfP/Zoco>

Organizadores

José L. Álvarez, José L. Arjona (Universidad de Huelva)
Rafael Corchuelo, David Ruiz (Universidad de Sevilla)

Ponentes

Carlos Guerrero, Carlos Juiz, Ramon Puigjaner (Universidad de las Islas Baleares).
Fernando de la Rosa, Rafael M. Gasca (Universidad de Sevilla). José M. Blázquez, Jesús
Arias, Vicente Luque, Luis Sánchez (Universidad Carlos III de Madrid). Carlos Rivero
(TB Solutions). Pablo Palacios (Universidad de Huelva). Francisco J. Pinto (Universidad
de la Coruña). Diana M. Sánchez, César J. Acuña, José M^a Cavero, Esperanza Marcos
(Universidad Rey Juan Carlos). Antonia M^a Reina, Miguel Toro, Jesús Torres
(Universidad de Sevilla).

Índice

Prólogo del Taller, 1

José L. Álvarez, José L. Arjona, Rafael Corchuelo, David Ruiz

Dynamic Web Fragment Architecture, 3

Carlos Guerrero, Carlos Juiz, Ramon Puigjaner

Extracción Automática de Redes Sociales, 12

Fernando de la Rosa, Rafael M. Gasca

Ordenación de Resultados en Consultas Semánticas sobre Bases de Datos, 22

José M. Blázquez, Jesús Arias, Vicente Luque, Luis Sánchez

An Advanced Query Capability Description Framework Implementation, 31

Carlos Rivero, David Ruiz

Acelerando el Desarrollo de Extractores de Información con SRV, 39

Pablo Palacios, José L. Arjona, José L. Álvarez

Uso del Recurso Lingüístico WordNet en la Expansión de Consultas, 45

Francisco J. Pinto

UML+OCL: Una Alternativa para Modelar Servicios Web Semánticos, 52

Diana M. Sánchez, César J. Acuña, José M^a Cavero, Esperanza Marcos

Integrando Aspectos en MWACSL, 62

Antonia M^a Reina, Miguel Toro, Jesús Torres

Prólogo del Taller

José L. Álvarez⁽¹⁾, José L. Arjona⁽¹⁾, Rafael Corchuelo⁽²⁾, David Ruiz⁽²⁾

⁽¹⁾ Universidad de Huelva. Dep. de Tecnologías de la Información.
Escuela Politécnica Superior. Ctra. Huelva-La Rábida. Palos de la Frontera 21071.
{alvarez,jose.arjona}@diesia.uhu.es

⁽²⁾ Universidad de Sevilla. Dep. de Lenguajes y Sistemas Informáticos
ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012.
{corchu,druiz}@us.es

1. Motivación

La Web es el mayor repositorio de conocimiento de la Humanidad y ha conseguido un enorme éxito gracias a que la información que ofrece está disponible a través de mecanismos de interacción amigables y en un formato fácil de entender por las personas. Esta es la razón por la que muchos autores hacen referencia a los sitios web como islas de datos amigables en la Web. Además, los Servicios Web y la Web Semántica están revolucionando este repositorio ya que sus tecnologías facilitan a los agentes software el acceso al mismo y abren las puertas a innumerables posibilidades de integración de información y aplicaciones.

El problema para usar estas tecnologías en sitios web ya existentes es que requieren una reingeniería que no siempre es viable (aplicaciones no desmantelables), lo que supone retos desde el punto de vista de la Ingeniería del Software.

El objetivo de Zoco'07/JISBD es ofrecer a los investigadores interesados en estos temas un punto de encuentro en el contexto de las Jornadas de Ingeniería del Software y Bases de Datos. Los trabajos presentados en el taller han sido seleccionados cuidadosamente con el objetivo de proporcionar la mayor diversidad posible de ideas útiles para la integración de aplicaciones web no desmantelables.

2. Trabajos presentados

El primer trabajo ha sido realizado por Guerrero, Juiz y Puigjaner, de la Universidad de las Islas Baleares, y presenta una propuesta para mejorar el cacheo de fragmentos de páginas web que tiene aplicación, por ejemplo, como una forma de mejorar el rendimiento en la integración de aplicaciones basadas en portlets.

El segundo trabajo ha sido realizado por de la Rosa y Gasca, de la Universidad de Sevilla, y presenta un algoritmo para la extracción de redes sociales de la web; estos resultados pueden ser de utilidad para resolver problemas de correferenciación de personas.

El tercer trabajo es de Blázquez, Arias, Luque y Sánchez, de la Universidad Carlos III de Madrid, y presenta una medida de similitud entre elementos de una taxonomía, lo que de nuevo puede ser de gran interés para resolver problemas de correferenciación.

Rivero, de TB Solutions, y Ruiz, de la Universidad de Sevilla, son los autores del cuarto trabajo, en el que se presenta una implementación de un framework para rellenar formularios automáticamente a partir de consultas en SQL.

El quinto trabajo ha sido realizado por Palacios, Arjona y Álvarez, de la Universidad de Huelva, y presenta algunos resultados que permiten mejorar la eficiencia del algoritmo FOIL en el contexto del extractor de información

2 *Prólogo del Taller*

SRV.

El sexto trabajo es de Pinto, de la Universidad de la Coruña, y en él se presenta y evalúa una propuesta para utilizar WordNet como complemento que permite desambiguar el significado de algunos términos a la hora de lanzar consultas en la web.

El séptimo trabajo es de Sánchez, Acuña, Caverro y Marcos, de la Universidad Rey Juan Carlos, y en él se presenta una propuesta que combina UML y OCL como alternativa para el modelado de servicios web semánticos.

Finalmente, el trabajo de Reina, Toro y Torres, de la Universidad de Sevilla, presenta una propuesta orientada a aspectos y dirigida por modelos para el desarrollo de aplicaciones web.

Dynamic Web Fragment Architecture

Carlos Guerrero, Carlos Juiz, Ramon Puigjaner

Universitat de les Illes Balears. Dep. de Ciències Matemàtiques i Informàtica.
Escola Politècnica Superior. Crta. Valldemossa, km 7'5. Palma de Mallorca 07122.
{carlos.guerrero,cjuiz,putxi}@uib.es

Abstract

This paper gives the details of a web caching architecture which uses ESI and web document fragments to improve the behavior of the web pages from a performance point of view. As the performance of the web page that use fragments depends on the state and parameters fixed in running time, the design of the fragments has to be dynamic and alterable at that phase. The fragment design of web pages could be change dynamically applying operations over the web documents fragments, the operations to apply will be determined using ontologies and a reasoner to analysis the behavior of the users and web server tiers for each fragment design. Changing web pages fragment design at running time generates integrations problems with the existing web applications which are not able to provide these fragments. The reutilization of the existing web applications will depend on the capacity to adapt them to an scenario where code fragment execution will be provided.

1. Introduction

Web servers has completely changed the way contents are delivered. One of the major changes introduced has been the traditional generation of the context in a dynamic way. At the beginning contents were static and the process at the server to delivered that content involves only the task or reading the static file. In web dynamic environments the process to deliver contents to users incurs: 1) user requests are dispatched to appropriate software

modules that service these requests, thereby producing network overhead; 2) these software modules determine which data to fetch and present, thereby producing process overhead; 3) the disk I/O management querying the back-end database produces data overhead, and finally; 4) the assembled data needs to be formatted and delivered to the browser, thereby producing cache overhead. In short, in the Web pages on-the-fly building process a higher number of activities or process take part than in the static web sites. Therefore, the dynamic workload and the computational cost are higher than the workload generated by a static web server. This is the reason why content caching has become an interesting research topic in web engineering [11].

Unfortunately, there are other more complex problems appearing in dynamic web caching. For instance, the ratio in which web pages change is usually higher than the static ones and moreover, these changes are usually small. These small changes produce the page invalidation in the cache (miss) and the corresponding request to the server is sent, generating the additional respective workload. Another common case in dynamic server is the huge amount of information that the web pages are sharing. This shared information has to be generated for each page even though it is identical.

Since the minimum caching unit in web servers is a full document, the problem of regenerating web pages, in every content change, turns reusing shared information out almost an impossible task. However, we may improve the way of the web pages are cached by reduc-

4 Dynamic Web Fragment Architecture

ing the caching units to fragments of pages. This technique avoids the regeneration of the whole page. One way to brake down pages into fragments is given by ESI (Edge Side Includes) [10].

Unfortunately, the design of the fragments to obtain the best performance of the pages request depends on dynamic parameters as content change ratio, content share ratio, user requests and network and server performance. That parameters can not be known in the development phase so they only can be gathered when the web application is running. If the target is to improve the performance of the system using a fragment design of the documents of the web application, it is necessary to gather the performance parameters and change the fragment design in a dynamic and automatic way.

In that paper a tool for a dynamic fragment design is proposed. That tool uses an ontological language to analyze the performance information gathered along user request. Using that information, the fragment design of the pages changes dynamically. Also a ontological language is used to analyze and represent the fragment design. Thus, in section 2 and 3 a brief introduction to ESI and ontological and Semantic Web is provided. In section 4 the proposed architecture and tools are presented. Conclusions and future work are presented at the end of the paper.

2. Edge Side Include

Edge Side Includes (ESI) is a simple markup language that web application developers may use to identify content fragments for web dynamic assembly. ESI also specifies a content invalidation protocol for transparent content management across ESI-compliant solutions, such as application servers and content delivery networks. The ability to assemble dynamic pages from individual page fragments means that only non-cacheable or expired fragments need to be fetched from the origin Web site, thereby lowering the need to retrieve complete pages and decreasing the load on the Web site's content generation infrastructure.



Figure 1: Example of ESI template page (white background) containing ESI fragments (gray background fragments: A,B,C,D,E,F).

The use of ESI maximizes its profit in scenarios where the response times are high in the server, in the network or in both cases. ESI can be use in the most layers of the web architecture: in a web cache proxy in the client side, in an inverse cache proxy in the server side, or in the edge of the network.

ESI accelerates dynamic Web-based applications by defining a simple markup language to describe cacheable and non-cacheable Web page components that can be aggregated, assembled and delivered. ESI enables Web pages to be broken down into fragments of different cacheability profiles. These fragments are maintained as separate elements in the application server's local cache and/or on the content delivery network. ESI page fragments are assembled into HTML pages when requested by end users. This means that much more dynamically generated content can be cached, then assembled and delivered when requested. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

The basic structure a content provider uses to create dynamic content in ESI is a template page containing HTML fragments (Figure 1). The template consists of common elements such as a logo, navigation bars, and other look-and-feel elements. The HTML fragments represent dynamic subsections of the

```

<table><tr><td>
  <esi:try>
    <esi:include src=
      "http://www.myserver.com/main.php">
    </esi:try>
  </td></td></table>

```

Figure 2: Example of ESI template code with the source of a fragment.

page. The template is the file associated with the URL the end user requests. It is marked-up with ESI language (Figure 2) that tells the cache server or delivery network to fetch and include the HTML fragments. The fragments themselves are HTML/ESI marked-up files containing discrete text or other objects.

The technique of breaking down pages into fragments does not always benefit the web performance. When a page does not share information with other pages there is no advantage of having cached fragments of the document. Thus, it is necessary to generate all the fragments and to assemble all them. In these cases the use of ESI produces an additional overload in comparison with the traditional way of generating web pages [2].

Moreover, breaking the web page down in a high number of parts could cause a worse response time because the main part of the fragments are not shared with other pages. Then, the changes in that page only affect to the template or to one fragment. On the contrary, by breaking the page down in a small number of parts also could cause a worse response time that the traditional way to cache pages. If fragments are too big, it is less likely to find them in other pages, and little changes could invalidate the main part of documents.

The way in which the document is fragmented takes a crucial importance from the performance point of view. But it is not as easy as make the higher possible number of fragments; it depends in the grade fragments are sharing information, the frequency of fragment updates and the workload needed to generate the fragment. Therefore, a fragment construction is more convenient as these three parameters are higher. However, this knowledge

could not be known by the web developer or web analyst during the software engineering process. Additionally, these three parameters are not constant along time. Thus, fragments have to be dynamically built using the information gathered in previous web requests.

The work presented in this paper is part of a research project to build a tool for analyzing these parameters and fragments of the different pages in order to obtain the optimal performance. These fragments would be made using the information gathered along requests, and would change dynamically so long as those parameters are changing.

3. Ontological languages

The Web Ontology Language (OWL) is a language for defining and instantiating Web ontologies [9]. An OWL ontology may include descriptions of classes, along with their related properties and instances. OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. It facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. OWL is seen as a major technology for the future implementation of a Semantic Web. It is playing an important role in an increasing number and range of applications, and is the focus of research into tools, reasoning techniques, formal foundations, etc.

OWL was designed to provide a common way to process the semantic content of web information. It was developed to augment the facilities for expressing semantics (meaning) provided by XML, RDF, and RDF-S. Consequently, it may be considered an evolution of these web languages in terms of its ability to represent machine-interpretable semantic content on the web. Since OWL is based on XML, OWL information can be easily exchanged between different types of computers using different operating systems, and application languages. Because the language is intended to be read by computer applications, it is some-

6 Dynamic Web Fragment Architecture

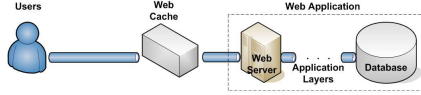


Figure 3: Web Caching Architecture.

times not considered to be human-readable, although this may be a tool issue. OWL is being used to create standards that provide a framework for asset management, enterprise integration, and data sharing on the Web.

4. Proposed Architecture

Typically web-based applications are based in tiered architecture. Usually in these applications the user interface, functional process logic (business rules), data storage and data access are developed and maintained as independent as is possible, most often on separate platforms.

Dynamic web sites commonly are developed using different layers to reduce the cost of maintenance and the idealistic layered application is three-tier web-sites:

- A front Web server serving static content.
- A middle dynamic content processing and generation level Application server.
- A back end Database, comprising both data sets and the Database management system or RDBMS software that manages and provides access to the data.

As the number of layers and interfaces between layers increase, the generation process also is bigger and computationally more expensive. To minimized the workload generated by each user request response, a cache layer could be placed between clients and server layers (Figure 3). Web caching also contributes to reduce bandwidth usage, server load, and user perceived lag. The caching tier stores generated content to avoid the load over the server tiers at next request of the same page.

As is presented in Section 2, caching could be improved reducing the minimum caching

unit [8]. Instead whole pages would be cached, if pages fragments are cached independently, the performance of dynamic webs could be improve. But the way that fragments are made, fragment design, is important to achieve a better performance that in whole pages caching scenarios. And also, the best fragment design, from the performance point of view, could change along time.

Figure 4 shows the proposed scenario to define pages fragment design in a dynamic way, using the data gathered about the pages and fragments behavior. The tier architecture is the same that in the Web Caching Architecture but the caching tier has four independently modules:

- *The caching module* that listens the user request, stores the temporally copies of the web documents and requests new web documents to the server side when is necessary.
- *The Behavior Knowledge Base* that has data about the performance of each user request and the behavior of each web server module, resource or document.
- *The Fragment Design Knowledge Base* that has the information about the fragments of each web document, the fragment organization and the state of each fragment, i.e. the fragment design.
- *The Reasoner* that uses the information in the Behavior Knowledge Base to change the Fragment Design trying to achieve a better performance behavior.

A complete explanation about each module and the changes on the Web Server Tiers is presented next. It would also be needed to change the Application Tier in the Web Server to serve fragments of document avoiding the change of the applications code.

4.1. Web Cache Tier

Traditional Web Cache Tier performs the tasks of listen users requests, analyze if those requests are store locally in the cache, in which

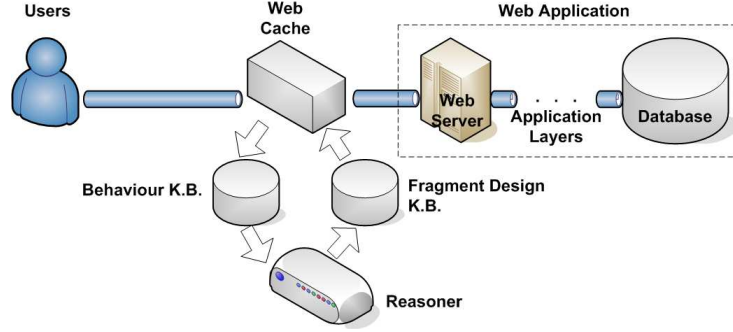


Figure 4: Web Caching Proposed Architecture.

case, the local copy is returned or, if the copy is not present, send the request to the server and updated the local version with the server response [12]. With the proposed architecture the Web Cache Tier performs more tasks, and it is only the highest level of the complete caching system (Figure 4). Levels below the Web Cache Tier store the data gathered by the Web Cache Tier and give it the information about the Fragment Design.

The process of gathering information about the behavior of the server site and the user requests is performed by that level. It's important to know the user profiles: pages most requested, navigation paths, thinking times, and so on. From the server site is important to know the response time and the size of each request, the changing ratio of the web pages contents and, finally, the sharing ratio between web pages contents. Those parameters are the most interesting to achieve the fragment design with the best performance.

As it is explained in section 4.2, the benefits of ESI depends on the sharing ratio between the cache stored contents and response contents by new request. That ratio depends on how usually is request each page, the contents shared between pages, and how often each page changes. Also is more important to improve pages with worst performance, or high server workload, than the fastest one, or pages that generate less workload.

A part from gathering behavior data, the Web Cache Tier requests the pages as it is indi-

cated by the levels below, the Fragment Design Knowledge Base. When a request is received, the first step is ask the page fragment design that determines the fragments needed to assemble the page. Only fragments that are not stored in the cache will be requested to the server. That fragment design changes along time depending on the gathered parameters.

4.2. Behavior Knowledge Base

The objective of the research work presented is create a tool that changes the fragment design of web pages using the user requests performance information gathered. Recent research works present studies about the way ambient environment respond dynamically using the information about the state of the system or environment [4, 5].

In that research works, ontological languages are used. Data about the state of the environment and the performance of the system is stored by instances of an ontology. The same idea is applied in the research work presented in that paper. The presented framework is a system where ontologies are used to managed the information about the state of the system (Fragment Design Knowledge Base) and the behavior of the user and the servers tiers (Behaviour Knowledge Base). A reasoner analyzes the information provided by the Behavior K.B. and applies operation to the instances of the Fragment Design K.B.

The important behavior data for the system

can be summed up in three groups: (a) user behavior; (b) server tiers performance and (c) content parameters [7, 6, 3, 1]. That parameters will be stored in an ontology with data and rules between that data. And the reasoner (Section 4.4) uses that data to apply operations to change the fragment design.

The data about user behavior gives us an idea about the behavior of the user (navigation paths, response times, and so on) when browses the web system. It is necessary stored the next parameters:

- *User request*, the resource or service requested by the user (URL), and the time of the request.
- *Session*, the identification of the user that make the request and the session in which the request is made.

The server performance is determined by parameters which gives an idea of the server side service. The list of interesting parameters for our architecture is:

- *Response time*, the server service time and the network time for each user request and for each fragment request generated by an user request.
- *Response size*, the size of the HTML code returned by the server (for fragments and pages).

Finally, the data about the content of the web pages has to give us an idea about the ratio in which contents change and the quantity of shared data between pages:

- *Share ratio*, the ratio in which the contents of the pages or fragment are share with other pages and how many pages have that contents.
- *Change ratio*, the ratio in which the contents of the pages or fragments change.

4.3. Fragment Design Knowledge Base

Using gathered information about the system behavior the fragment design of the pages

would be changed. It is necessary to maintain information about how pages have to be fragmented. An ontology is used for this propose. For each resource of the web system is known the fragments that form them and how to request that fragments to the origin server. In the same way, each fragment could be defined as an union of other fragments. All that information is characterized by the ontology. But the design (the definition of the fragments and pages) is continuously changing. Changes in the behavior of the resources produce change in the way page fragments are organized.

But not all the possible fragments of the pages are useful. We could not have a better performance braking down any part of the pages, especially parts which HTML code is generated by indivisible process in the server. Indivisible process are loops, if sentence, functions calls, and so on. For example, if we generate all the rows of a table with a loop structure, broke down the HTML code of the table in two parts do not suppose any profit, because each time we need a part of the table, all the loop has to be execute.

So the beginning and the end of the interesting fragments of the HTML code is only place in some determined places as are the beginning and the end of structures as if sentences, loops, functions calls, and so on. Thus the Fragment Design Knowledge Base does not have information about the server application codes, the HTML has to be extended. So the server applications has to add some extra tags to indicate which are the candidate fragments. Section 4.5 explains how to avoid make changes in the way programmers work giving the responsibility of add those tags to the web server instead the programmers.

ESI (Edge Side Includes) is used as an extension to HTML to define pages as a congregation of different parts which could be cache and requested independently. The basic idea is to Extend ESI to an ontology language where operations (defined in section 4.4) could be done over fragments.

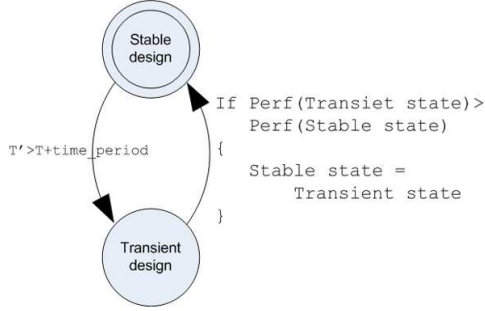


Figure 5: Pages state diagram.

4.4. Reasoner

The reasoner uses the information about the behavior of the clients, server tiers and server resources to determine the best distribution of the fragment of each resource (fragment design). But not all the arbitrary fragment are possible to improve the performance of the system. As it is explained over, only the HTML code generated by the same coding structures is a profitable fragment. Later, the reasoner evaluates the design or state of each page. It analyzes the behavior of the resource with the current design (stable design). Apply some operations over the page state so the design and fragment structure changes (transient design). After a period of time, it evaluates the behavior of that design. If that new design obtain a better performance, it will be maintained from now as the stable design, if not, the initial design or state is restored. After a period of time, the process is repeated. The period of time between different evaluations of possible new stable states is dynamic. Each time a stable design is restored after an evaluation period, the period of time for re-evaluation is increased. Figure 5 shows a diagram of the states or design changes.

To create a new state or design (transient state) the reasoner applies operations to the fragments and the pages. There are four different possible operations:

- *Break down*, Part of a fragment is created as an independently fragment (Fig-

ure 6.a).

- *Include*, A fragment is include in its father fragment (Figure 6.b).
- *Split*, A fragment is divided in two independently fragments (Figure 6.c).
- *Join*, two contiguous fragments are unified in only one fragment (Figure 6.d).

The web pages initial state would be the highest fragmented state. After periods of time applying operations and evaluating the web pages behavior a stable state would be reached. This transient state would take much time but after that initial time, changes in the state would be only when the behavior of the web pages or clients changes.

The operations the reasoners apply over the fragment are chosen according to share ratio, update ratio, workload associated to the generation of the fragment and number of requests of the fragment.

It is interesting to isolate fragments with low or null update ratio. If fragments are usually updated it is necessary to invalidate the copy of fragment in the cache system. If a fragment is the sum up of two fragments and one has a high update ratio and the other a low ratio, each time the first one is updated all the content has to be invalidated. So it would be interesting to have these fragments split in two to reduce the amount of data regenerated in each invalidation. In the same way, when two fragments have similar update ratio it could be considered as one fragment.

Also it is important to isolate fragments with similar share ratio. Isolating contents usually shared between pages, each time one of these pages are requested a hit over the isolated fragment would be occurred. So it is necessary more effort in the fragment design over contents with a high share ratio than over the low ones. But the number of times a fragment it is requested not only depends on the share ratio. The times a page is requested also take importance. If some contents have a high share ratio but the pages which have the contents are not usually requested these fragment would be generated a low number of times.

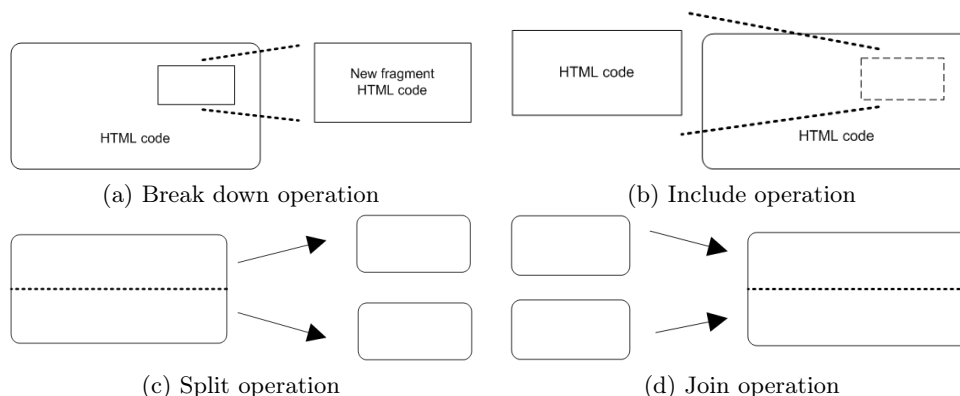


Figure 6: Reasoner operations.

Thereby, the contents in which the highest effort to make an accurate fragment design has to be done are the fragments with a high share ratio and a high number of requests.

Finally, the workload associated to the generation of the fragment take importance in the design process. It is more important to accurately the fragment design of contents that suppose a higher resource consume. The reasoner has to take into account that four parameters gathered from each fragment and takes the decision about the operation to take over the fragments.

4.5. Web Server Tiers

For using that new architecture is needed to generate parts of the pages or resources independently. That will involves a change in the way programmers build applications and changes all the existing applications. That is not the desired situation. Instead changing the applications and the way programmers work, the solution is change the web server tiers.

Web server tier would be able to execute only the applications code which generates the different fragments of the pages. That involves the detection and identification of the fragments, as well as the partial execution of web resources. The detection and identification of the different fragments of a web page involves and analysis of the source code of the applications that generate that web pages. Not all

possible combinations of fragments are useful. Applications have code structures that have to be execute atomically, it is not possible to execute a part of that structure. Examples of that are while/for sentences, if sentences, basic code blocks, and so on. To detect all the interesting fragment requires and analysis of the source code to detect all that structures. Also a study of which are those interesting structure is needed.

Once the code structures are detected, an identifier is assigned them. To differentiate the HTML code fragments generated the identifier has to sum up the variables and states which condition the execution and the HTML generated. For instance, if a for sentence depends on the value of variable *A*, the identifier of the fragment has to contain the value of *A* for the different executions. If the execution of two instances has the same value for *A*, the generated fragment is the same. If the code structure which generate the fragment depends on more than one variable or condition, they all must be part of the identifier.

5. Conclusions

An architecture to improve web performance has been proposed. That architecture is based on the idea of improve web performance caching fragments of documents instead the whole documents. In that way it

could be taken profit from the shared information between pages and could be avoid the regeneration of the whole page each time a little change is done in the web contents.

As the parameters that condition the performance and behavior of the system are not known in the development phase and could change along time, the fragment design has to be determined in running time. An ontological system is proposed to analysis the behavior of the system and make changes to the fragments trying to improve performance.

Future work includes creating the ontological system to analyze the system behavior and change fragment design, as well as the detection of the basics fragments blocks in the web server tiers and how to execute only the part of the services that generate that basics fragments blocks.

References

- [1] Pierre Baldi, Paolo Frasconi, and Padhraic Smyth. *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. Wiley, 2003.
- [2] Carlos Guerrero, Carlos Juiz, and Ramon Puigjaner. The applicability of balanced esi for web caching. *3rd International Conference on Web Information Systems and Technologies*, 2007.
- [3] Patrick Killelea. *Web Performance Tuning*. O'Reilly & Associates, 2002.
- [4] Isaac Lera, Carlos Juiz, and Ramon Puigjaner. Performance-related ontologies and semantic web applications for on-line performance assessment intelligent systems. *Sci. Comput. Program.*, 61(1):27–37, 2006.
- [5] Isaac Lera, Pere P. Sancho, Carlos Juiz, Ramon Puigjaner, Joachim Zottl, and Gunter Haring. Performance assessment of intelligent distributed systems through software performance ontology engineering (spoe). *Software Quality Control*, 15(1):53–67, 2007.
- [6] Daniel A. Menasce and Virgilio Almeida. *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall, 2001.
- [7] Daniel A. Menasce and A. F. Almeida. *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall, 2000.
- [8] Michael Rabinovich, Zhen Xiao, and Amit Aggarwal. Computing on the edge: a platform for replicating internet applications. In *Proceedings of the 8th international workshop on Web content caching and distribution*, pages 57–77, Norwell, MA, USA, 2004. Kluwer.
- [9] Michael K. Smith, Chris Welty, and Deborah L. McGuinness. Owl web ontology language guide. W3c recommendation, World Wide Web Consortium, 2004.
- [10] Mark Tsimelzon, Bill Weihl, Joseph Chung, and Dan Frantz. Esi language specification 1.0. W3c recommendation, World Wide Web Consortium, 2001.
- [11] A. Wierzbicki. Models for internet cache location. In *Proceeding of the 7th International Workshop on Web Content Caching and Distribution (WCW)*, 2002., 2002.
- [12] Craig E. Wills and Mikhail Mikhailov. Examining the cacheability of user-requested Web resources. In *Proceedings of the 4th International Web Caching Workshop*, April 1999.

Extracción Automática de Redes Sociales

Fernando de la Rosa, Rafael M. Gasca

Universidad de Sevilla. Dep. de Lenguajes y Sistemas Informáticos.
ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012.
{ffrosat,gasca}@lsi.us.es

Resumen

En este trabajo se presenta un proceso para extraer de forma automática la comunidad científica que trabajan sobre un determinado tópico de interés. El proceso se basa en el análisis de las coocurrencias de las direcciones de correo y permite determinar los centros de interés de la comunidad. De esta forma se define un marco de trabajo para integrar distintas fuentes bibliográficas accesibles desde Internet a partir de los resultados ofrecidos por los motores de búsqueda. A modo ilustrativo se aplica esta técnica para extraer la red social de la comunidad de las JISBD.

1. Introducción

Una de las consecuencias que provoca la expansión de la red Internet es que la cantidad de información pública crece exponencialmente. Esta información se encuentra almacenada en un conjunto de fuentes heterogeneas enlazadas entre sí. Dentro de Internet los motores de búsquedas juegan un papel fundamental en la búsqueda de información, pero el enfoque generalista de estos sistemas no permite integrar las distintas fuentes indexadas. En este contexto, la integración de fuentes de información heterogénea, el análisis de las coocurrencias se hace imprescindible.

En este trabajo se propone un *proceso para extraer de forma automática la comunidad científica que trabajan sobre un determinado tópico de interés*. Como se verá el proceso rea-

liza de forma sistemática consultas especialmente diseñadas a los motores de búsqueda y mediante un posterior análisis de las respuestas recibidas extrae la red objeto de interés. Analizando la información extraída junto con los resultados ofrecidos por los motores de búsqueda también será posible determinar los centros de interés de dicha comunidad. De esta forma se define un marco de trabajo para la integración de fuentes bibliográficas.

El trabajo se distribuye en las siguientes secciones. En la primera sección se presenta una revisión de los trabajos relacionados y se compara estos trabajos con la propuesta realizada. En las siguientes secciones se presenta la notación y el ámbito de aplicación de la propuesta. Para posteriormente concretar el proceso automático de extracción de redes sociales en las siguientes secciones: extracción de direcciones de correo electrónico, transformación de las consultas en redes sociales y orientación de la red a unos tópicos. A modo ilustrativo se aplica el proceso para extraer la red social de la comunidad de las JISBD (se puede acceder a la versión on-line de los experimentos en [16]). Finaliza este trabajo con las conclusiones, donde además se expondrán los trabajos futuros.

2. Trabajos relacionados

En trabajos previos se han definido procesos de extracción de redes sociales para diversas fuentes de información web: motores de búsquedas [7, 11, 13], chats [15], DBLP [3], archi-

vos FOAF [13, 14], SourceForge [1], listas de distribución [2], etc. La revisión del estado del arte que se realiza en este trabajo se restringe al análisis de los trabajos que utilizan motores de búsquedas para extraer redes sociales.

Entre los sistemas que se apoyan en los motores de búsquedas podemos encontrar el sistema REFERRAL WEB [7] que fue diseñado para extraer redes sociales a partir de consultas al motor Altavista, la red obtenida se centra en una persona determinada (red egocéntrica). Para ello solo necesita conocer el nombre del ego. Y mediante un reconocedor de entidades el sistema extrae una lista de personas con las que se relaciona. Para medir la relevancia de la relación entre X e Y (la variable Y contiene cualquiera de los nombres de la lista de personas relacionadas con X) utiliza el coeficiente Jaccard [6]. El proceso descrito puede repetirse recursivamente. En la Figura 1 se puede observar un ejemplo de cómo calcular el coeficiente Jaccard, medida que llamaremos afinidad.

Recientemente se han presentado dos sistemas POLYPHONET [11, 12] y FLINK [13, 14] que generan la red social de una comunidad científica partiendo de una lista con los nombres de sus miembros. El algoritmo básico de ambos sistemas construye la matriz de afinidad realizando consultas para cada pareja X e Y (X e Y son dos nombres distintos de la lista). Ambos sistemas utilizan un umbral de corte para determinar cuando las relaciones son significativas. A continuación presentamos los inconvenientes de estos sistemas:

- **Clasificación de relaciones:** Los índices de afinidad calculados son de difícil interpretación. La co-ocurrencia de nombres en las páginas indexadas puede deberse a múltiples factores: publicación conjunta, simple casualidad, participación en un mismo evento (comité de programa), trabajos referenciados en un mismo artículo, etc.
- **Nombres ambiguos:** Al utilizar nombres de personas en las consultas se está agregando ambigüedad a sus resultados, la probabilidad de que el nombre haga referencia a más de una personas es alta.

- **Variedad de nombres:** Suele suceder que se haga referencia a una misma persona con varios nombres, por ejemplo “Rafael Martínez Gasca” o “R. M. Gasca”.
- **Escalabilidad:** Los sistemas que utilizan listas cerradas de nombres no son escalables. Para conseguir la escalabilidad de estos sistemas [7, 12] proponen utilizar reconocedores de entidades para extraer nuevos nombres a partir de las referencias devueltas por el motor de búsqueda.
- **Complejidad:** En el peor caso los algoritmos revisados son $O(N^2)$, donde N es el número de miembros de la red, para cada pareja de actores deben realizarse una consulta. Esto es un problema relevante teniendo en cuenta que las licencias de uso que distribuyen los motores de búsqueda pueden tener limitaciones. Por ejemplo, Google no permite realizar más de 1000 consultas diarias (para completar una red de 500 actores serían necesario 125 días). La implementación de algoritmos escalables [12] ha permitido reducir la complejidad respecto del número de consultas necesarias para completar la red. En [12] se propone un algoritmo de $O(N)$ (según datos del autor para 503 actores es necesario 19.852 consultas, 20 días).

Los trabajos descritos utilizan listas de nombres para extraer redes sociales. En contraposición existe la posibilidad de sustituir las listas de nombres por direcciones de correo. En [8] se describe como extraer relaciones sociales entre los usuarios de un dominio específico. El algoritmo descrito es de $O(N^2)$ y es utilizado para realizar ataques de ingeniería social en el ámbito de la seguridad de los sistemas informáticos.

El algoritmo de extracción propuesto utiliza direcciones de correo y mejora sustancialmente la complejidad de los algoritmos revisados. Además utiliza un modelo de consulta que lo hace robusto ante la ambigüedad y permite una clara interpretación de las relaciones extraídas sin necesidad de utilizar algoritmos de aprendizaje. A diferencia del trabajo anterior

The figure consists of four panels showing search results from a search engine. Each panel has a search bar and a 'Búsqueda' button.

- Panel 1 (Top Left):** Search for "R. M. Gasca" and "C. del Valle". Results show 119 for "R. M. Gasca" and 410 for "C. del Valle". A formula calculates affinity: $Afinidad = 119 / (410 + 10900 - 119) = 0.0106335$. Below, a search for "R. M. Gasca" "C. del Valle" shows 12600 results, with a formula: $Afinidad = 119 / 12600 = 0.0094$.
- Panel 2 (Top Middle):** Search for "C. del Valle". Results show 10,900. A blue arrow points down to a search for "C. del Valle" Scheduling, showing 96 results. Below, a search for "R. M. Gasca" "C. del Valle" Diagnosis shows 96 results. A formula calculates affinity: $Afinidad = 96 / (255 + 207 - 96) = 0.262295$.
- Panel 3 (Top Right):** Search for "R. M. Gasca" "C. del Valle" (Diagnosis | Scheduling). Results show 108. Below, a search for "R. M. Gasca" Diagnosis shows 207 results. A formula calculates affinity: $Afinidad = 108 / (207 + 96 - 108) = 0.5538461$.
- Panel 4 (Bottom):** Search for "R. M. Gasca" "C. del Valle". Results show 119. Below, a search for "C. del Valle" "R. M. Gasca" shows 2 results. The word "Dance" is displayed at the bottom.

Figura 1: Ejemplos de distintos usos de los resultados ofrecidos por un motor de búsqueda.

[4], el algoritmo propuesto, no necesita una lista inicial de direcciones de correo y utiliza heurísticas que permiten dirigir la construcción de la red social en función de la temática y de la importancia de los miembros de la red.

3. Notación

A lo largo de este trabajo modelaremos las redes sociales como un grafo no dirigido que denotaremos como G . Los nodos de G se corresponden con los actores de la red y los lazos entre los actores son representados mediante las aristas de G . Denotaremos:

- G_N como el conjunto de nodos de G
- G_E como el conjunto aristas del grafo G
- $n \in G_N$ indica que n es un nodo de G
- $e \in G_E$ indica que e es una arista de G . De forma alternativa se representa e como (n_1, n_2) , siendo n_1 y n_2 dos nodos de G .

Tanto los nodos como las aristas podrán disponer de atributos, se denota con n_x y e_x al atributo x de n y e respectivamente, donde $n \in G_N$ y $e \in G_E$. A continuación se presentan los atributos utilizados:

- n_{hits} acumulador de n
- n_{email} correo electrónico de n
- n_{domain} dominio del correo de n

- n_{login} cuenta del correo de n

- e_{hits} acumulador de e

Para realizar transformaciones sobre los grafos se definen las primitivas:

Nodo(email, G): devuelve el nodo $n \in G_N$ cuyo atributo n_{email} sea igual al parámetro email. Si no existe, crea un nodo asignando el valor cero y el parámetro email a los atributos hits y email del nodo respectivamente.

Arista(n_1, n_2, G): devuelve la arista $(n_1, n_2) \in G_E$, si no existe la crea y asigna el valor cero al atributo hits de la arista.

4. Ámbito de aplicación

El proceso propuesto en este trabajo es susceptible de aplicarse a cualquier sistema KWIC (Key Words in Context) [8]. Típicamente estos sistemas indexan un corpus de documentos y permiten realizar consultas sobre el corpus utilizando palabras claves. La característica de los sistemas KWIC es que almacenan las palabras claves junto con su contexto, que suelen ser las N palabras a la izquierda y a la derecha. Por tanto, cuando se realiza una consulta muestran los documentos que contienen las palabras claves utilizadas en las consultas junto a sus contextos. Esto permite al usuario del sistema evaluar la importancia del documento sin necesidad de explorarlo.



Figura 2: Ejemplo de consulta al motor de búsqueda Google. En negrita las palabras claves utilizadas y alrededor su contexto.

Con objeto de poder comparar los resultados obtenidos con otras propuestas de extracción de redes sociales a través de la Web en este trabajo se utiliza como sistema de referencia el motor de búsqueda Google. Este motor está compuesto por un conjunto de agentes de búsquedas o crawlers que rastrean la Web e indexan los documentos que encuentran a su paso. Posteriormente mediante el uso de palabras claves permite consultar los documentos indexados junto a los contextos de las palabras claves (Figura 2).

5. Extracción de e-mails

Para iniciar el proceso extracción será necesario disponer de un conjunto de direcciones de correos. Nos referiremos a este conjunto como conjunto de semillas, S . Se pueden definir diferentes procesos para extraer direcciones de correos de la Web. Estos procesos suelen utilizar un crawler para visitar páginas web y extraen de las páginas direcciones de correos utilizando expresiones regulares. Para indicar al crawler que páginas deben visitar se suele realizar consultas a motores de búsquedas o se restringen la navegación a unos dominios específicos. Por ejemplo, para encontrar direcciones en el dominio “us.es” se puede:

- Navegar por las páginas del sitio

“www.us.es” y extraer las direcciones de correo de las etiquetas <a>

- Otra posibilidad, es realizar la consulta “us.es” -“www.us.es” y analizar mediante expresiones regulares los resultados para extraer direcciones de correos.

La extracción de direcciones de correo dirigidas a un dominio específico no resulta útil en el ámbito de este trabajo. Por eso, en la Figura 4 se define el proceso ExtracciónDeSemillas. Este proceso recibe dos parámetros umbral y W. Y devuelve un conjunto de direcciones de correo extraídas de la Web con un grado alto de asociación al tópico W. Para ello:

- Realiza consultas con el tópico W.
- Visita los documentos web ofrecidos por el motor de búsqueda.
- Extrae los correos de las etiquetas <a> de los documentos visitados.
- Y finalmente, para cada dirección de correo, comprueba mediante una consulta “<correo>” “<tópico>” el grado de asociación de la dirección de correo al tópico.

El proceso considera un grado de asociación alto si el número de documentos encontrados por la consulta anterior supera un umbral y utiliza las siguientes primitivas:

ExtractLinks: consulta al buscador y extrae los enlaces de las respuestas.

ExtractHits: extrae el número de enlaces recuperados en una consulta.

6. Expansión de las redes sociales

En el apartado anterior se propone un algoritmo para extraer direcciones de correo con un alto grado de asociación con el tópico. En este apartado se presenta el algoritmo ExpandeNodos y en la Figura 4 se formaliza el proceso. El algoritmo recibe dos parámetros: el grafo G que representa una red social y un conjunto $S \in G_N$. Como resultado se obtiene un nuevo grafo G' . El grafo G' será el grafo G expandido con nuevas relaciones extraídas mediante

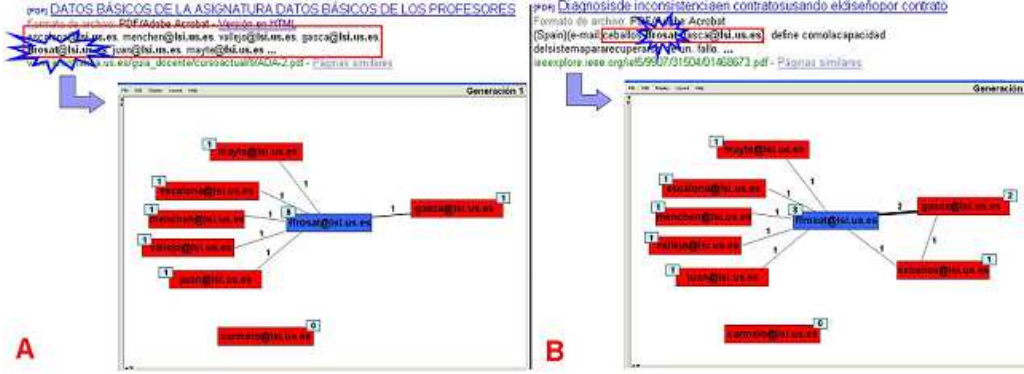


Figura 3: Transformación de los contextos de la consulta mostrada en la Figura 2 en una red social. Los nodos rojos son los actores añadidos al sociograma y los azules los nodos semillas. En cuadrados celestes los contadores asociados.

consultas a motores de búsqueda. Las consultas se construyen con las direcciones de correo de los nodos de S . Para cada n_{email} , donde $n \in S$, el proceso crea una consulta utilizando el esquema:

“<cuenta>” “<dominio>” filetype:pdf

Por ejemplo, la cuenta de la dirección `ffrosat@lsi.us.es` será `ffrosat` y el dominio `lsi.us.es`, por tanto la consulta realizada es:

“ffrosat” “lsi.us.es” filetype:pdf

Estas consultas se componen de dos partes: la parte 1) “cuenta” “dominio” fuerza al sistema a buscar los contextos en que aparece la dirección de correo 2) y la parte `filetype:pdf`, restringe la búsqueda a los documentos pdf (en la Figura 2 se muestra un ejemplo). Una vez obtenido el resultado de la consulta se analizan los contextos mediante expresiones regulares. Se analizan solo los contextos donde aparece la dirección n_{email} . Cada nueva dirección de correo que aparezca en el contexto formará parte de la red social como un nuevo nodo n' . Y se añadirá un lazo entre n y n' . Los nodos y los lazos están asociados con un contador, n_{hits} , que representará su importancia dentro de la red. Si alguna dirección vuelve a repetirse en algún contexto los contadores correspondientes se incrementarán en una unidad. Podemos

ver un ejemplo en Figura 3A. En los contextos pueden aparecer algunas expresiones de correo especiales como por ejemplo:

{ceballos,ffrosat,gasca}@lsi.us.es

En estos casos la dirección de correo forma un cluster donde todos se relacionan con todos, podemos ver un ejemplo de la transformación en la Figura 3.B.

Si bien es cierto que una persona puede tener más de una dirección de correo (por ejemplo, la personal y la institucional), también es cierto que una dirección de correo suele identificar a una única persona. Esto permite eliminar los problemas de ambigüedad que presentan otras metodologías, aunque no los problemas de variedad. Como veremos la utilización de este algoritmo permitirá la escalabilidad del proceso.

Respecto a la formalización del algoritmo el operador $emails \times emails$ representa el producto cartesiano de los elementos del conjunto $emails$ y el operador $em_1 < em_2$ establece una relación de orden entre dichos elementos. También se definen las siguientes primitivas:

ExtractContexts: devuelve un conjunto con los contextos extraídos de una consulta de tipo “cuenta” “dominio” filetype:pdf.

ExtractEmails: devuelve un conjunto con las direcciones de correo electrónico que encuentra

en el contexto c .

IsCluster: tras analizar el contexto c devuelve verdadero si las direcciones de correo extraídas de c forman parte de un cluster y falso en caso contrario.

7. Extracción dirigida por tópicos

Este proceso iterativo mantiene en una cola de prioridad, `queue_seeds`, los nodos del grafo G que son susceptibles de ser expandidos. Inicialmente la cola se inicializa con las semillas extraídas por el proceso ExtracciónDeSemillas. En cada iteración extrae un nodo, $n \in G_N$, de la cima de la cola y comprueba el grado de asociación de la cuenta de correo del nodo, n_{email} , al tópico. Si el grado de asociación supera un determinado umbral expande G con los nodos vecinos del nodo n . Para realizar la expansión de G aplica el ProcesoDeExpansión sobre el nodo n . La iteración se repite hasta que la cola `queue_seeds` esté vacía o la primitiva `endProcess` indique lo contrario. Esta primitiva se utiliza para limitar el número de páginas visitadas o nodos de G .

Tras finalizar la expansión de un nodo n se añade a `queue_seeds` los nuevos nodos que han aparecido en G . También existe la posibilidad de reordenar `queue_seeds` para maximizar la importancia de los nodos seleccionados en el proceso iterativo. El proceso `reorder_queue` es el encargado de realizar esta tarea. Este primitiva permite calcular distintas medidas para reordenar los nodos en función de su importancia: el grado, el pagerank ó la intermediación.

8. Experimentos

Para ilustrar la técnica descrita se ha utilizado el tópico “JISBD” para extraer la red social de la comunidad de la JISBD. Utilizando es tópico con el algoritmo ExtracciónDeSemillas se extrajeron 289 direcciones de correos relacionados con esta comunidad. Tras la expansión de la red con el ProcesoDeExpansión se amplió la red con 1.239 nodos y 1.941 relaciones. De estos nodos 430 superaron el umbral mínimo y fueron necesarias 2.371 consultas al motor de

búsqueda para generar la red. En la figura 5 se pueden ver los resultados.

Posteriormente se analizaron las consultas devueltas por el motor de búsqueda para los 430 nodos que superaron el umbral de asociación y se obtuvo la red léxica que se puede observar en la figura 6. Los resultados de la red léxica confirman la alta representatividad de la red extraída, ya que se ajusta a los tópicos de la comunidad de las JISBD. El proceso de construcción de la red léxica se basa en el análisis de las frecuencias de las palabras, la utilización de listas de palabras prohibidas y la búsqueda de colocaciones [10]. Los resultados obtenidos están disponibles on-line en [16].

9. Conclusión

En comparación con los trabajos revisados se aportan las siguientes mejoras:

- El proceso es totalmente automático.
- Al utilizar direcciones de correo electrónico se disminuye sensiblemente los problemas de ambigüedad. Aunque su uso, puede estar limitado en algunos casos por las técnicas antispan.
- El coste computacional del algoritmo es lineal respecto al número de actores de la red y mejora la constante multiplicativa de la mejor propuesta realizada hasta el momento. Según los datos aportados en [12] para 503 actores necesita 19.852 consultas, con esta propuesta serían necesarias 1.006 consultas.

El inconveniente del uso de esta técnica de extracción radica en la selección del término utilizado. La ambigüedad de éste puede influir negativamente en la información obtenida (por ejemplo la palabra *java* puede ser un lenguaje de programación pero también puede interpretarse como una isla). Este problema se puede minimizar diseñando consultas que excluyan palabras claves, como por ejemplo: *java AND NOT isla*.

Como trabajos futuros se plantea implantar una fase de clareado de datos (data clearing)

para eliminar las posibles variedades (direcciones de correo asociadas a una misma persona) [5] y procesos para refinar los lazos de la red.

Finalmente, como aplicación práctica, se ha implementado una herramienta disponible online en [16], que visualiza los resultados de los experimentos y los utiliza para precalcular consultas. De esta forma se aporta una perspectiva social a la búsqueda de información a través de una interfaz entre un usuario y un motor de búsqueda.

Referencias

- [1] Crowston, K. and Howison, J. (2004). *The social structure of free and open source software development*.
- [2] Culotta, A.; Bekkerman, R. and McCallum A. (2004). *Extracting Social Networks and Contact Information from Email and the Web*, Conference on Email and Spam.
- [3] De la Rosa T., F.; Pozo, S. and Gasca, R. M. (2005). *Análisis y Visualización de Comunidades Científicas con Información Extraída de la Web*, IEEE Latin America Transactions. ISSN: 1548-0992.
- [4] De la Rosa T. and Gasca, R. M. (2007). *Sistemas de Inteligencia Web basados en Redes Sociales*, Revista Hispana para el Análisis de Redes Sociales, Vol. 12. ISSN: 1579-0185.
- [5] Elmagarmid, A. K.; Ipeirotis, P. G. and Verykios, V. S. (2007) *Duplicate record detection: A survey*. IEEE Transactions on Knowledge and Data Engineering, 19(1):1-16.
- [6] Jaccard P (1901). *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*, Bull Soc Vaudoise Sci Nat 37:547-579.
- [7] Kautz, H.; Selman, B. and Shah, M. (1997). *The hidden Web*, AI magazine, 18(2):27-35
- [8] Long, J. *The Google Hacker's Guide*. Retrieved, 2005. ISBN: 1-59749-176-4.
- [9] Luhn, H. P. (1959). *Keyword-in-Context Index for Technical Literature (KWIC Index)*, Yorktown Heights, N. Y.: IBM.
- [10] Mason, O. (2001), *Programming for Corpus Linguistics*. Edinburgh: Edinburgh University Press. ISBN: 0-7486-1407-9.
- [11] Matsuo, Y.; Tomobe, H.; Hasida, K. and Ishizuka, M. (2003). *Mining Social Network of Conference Participants from the Web*, In Proceedings of the International Conference on Web Intelligence, 190-194.
- [12] Matsuo, Y.; Mori, J.; Hamasaki, M; Takeda, H.; Nishimura, T.; Hashida, K. and Ishizuka, M. (2006). *Polyphonet: An advanced social network extraction system*, In Proceedings of WWW2006.
- [13] Mika, P. (2004). *Bootstrapping the FOAF-Web: An Experiment in Social Network Mining*, Proc. 1st Workshop Friend of a Friend, Social Networking and the Semantic Web.
- [14] Mika, P. (2005). *Flink: Semantic web technology for the extraction and analysis of social networks*, Journal of Web Semantics, 3(2).
- [15] Mutton, P. (2004). *Inferring and Visualizing Social Networks on Internet Relay Chat*, InfoVis, Austin, TX, 35-43.
- [16] <http://www.lsi.us.es/~ffrosat/index.php/Ffrosat/MapaListaJISBD2005Es>

```

fun ExtraeSemillas(umbral, W)
  seeds ← {}
  email_set ← {}
  url_set ← ExtractLinks('W')
  foreach url ∈ url_set do
    [ email_set ← email_set ∪ ExtraeEmails(url)
  foreach EMAIL ∈ email_set do
    [ impacto ← ExtractHits('EMAIL ' + W)
    if (impacto > umbral)
    [ seeds ← seeds ∪ { EMAIL }
  return seeds

fun ExpandeNodos(G, S)
  G' ← G
  foreach s ∈ S do
    [ <L, D> ← <s_login, s_domain>
    contexts ← ExtractContexts('L D filetype:pdf')
    foreach c ∈ contexts
    [ emails ← ExtractEmails(c)
    if (s_email ∈ emails)
    [ if (IsCluster(c))
    [ foreach <em1, em2> ∈ { emails x emails } | em1 < em2 do
    [ [ <n1, n2, e> ← <Nodo(em1, G'), Nodo(em2, G'), Arista(n1, n2, G') >
    [ [ <n1_hits, n2_hits, e_hits> ← <n1_hits + 1, n2_hits + 1, e_hits + 1>
    else
    [ foreach em ∈ {emails ∈ s_email} do
    [ [ <n, e> ← <Nodo(em, G'), Arista(n, s, G')>
    [ [ <s_hits, n_hits, e_hits> ← <s_hits + 1, n_hits + 1, e_hits + 1>
  return G'

fun ExtraeRedSocial(umbral1, umbral2, W)
  initProcess()
  S ← ExtraeSemillas(umbral2, W)
  foreach email ∈ S do
    [ enqueue(queue_seeds, Nodo(email, G))
  <GN, GE> ← <{}, {}>
  while (not isempty(queue_seeds) and not endProcess())
  [ n ← dequeue(queue_seeds)
  EMAIL ← n_email
  hits ← ExtractHits(EMAIL + W)
  if (hits > umbral1)
  [ G' ← ExpandeNodos(G, { s })
  new_nodes ← G' - GN
  foreach nn ∈ new_nodes
  [ enqueue(queue_seeds, nn)
  [ reorder_queue(queue_seeds)
  [ G ← G'
  return G

fun reorder_queue(queue)
  sort queue by hits

```

Figura 4: Algoritmo de extracción de redes sociales

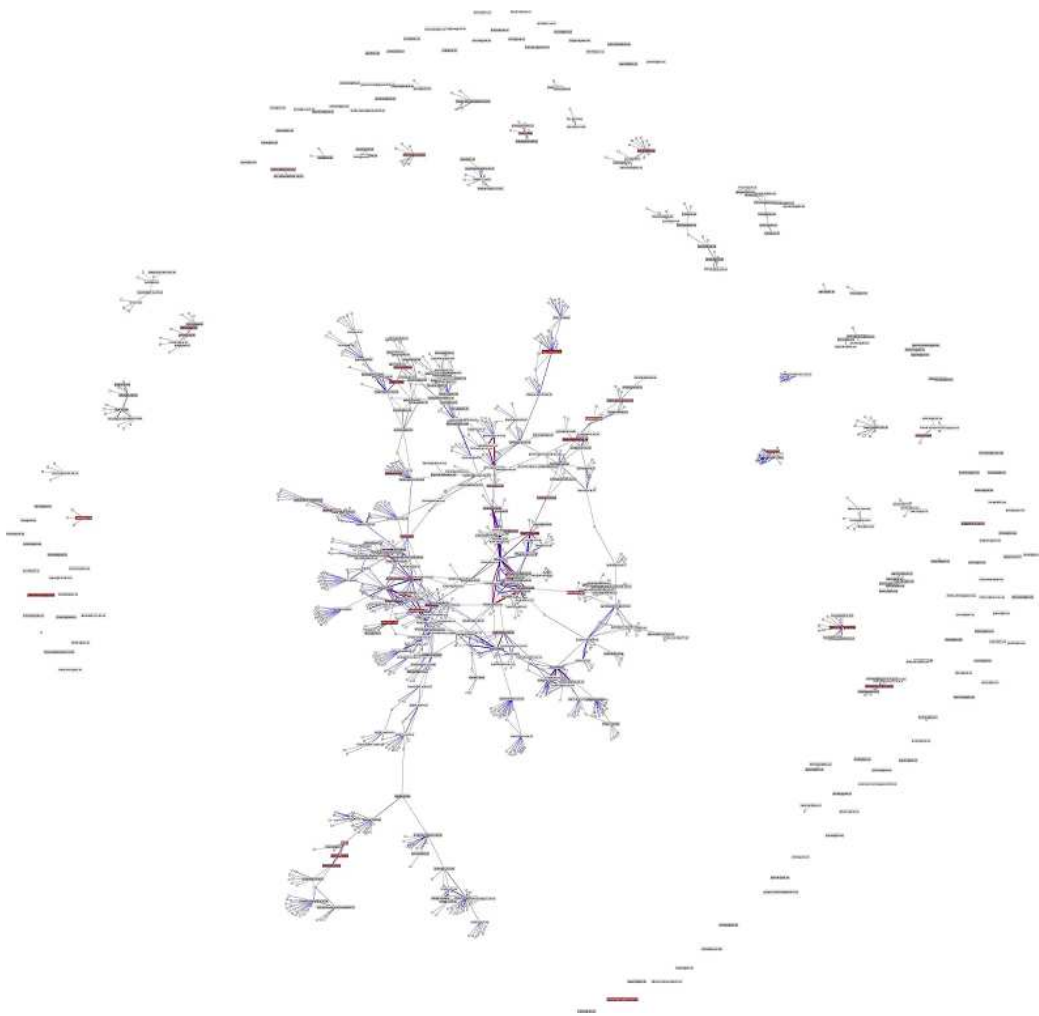


Figura 5: Comunidad de las JISBD extraída en los experimentos

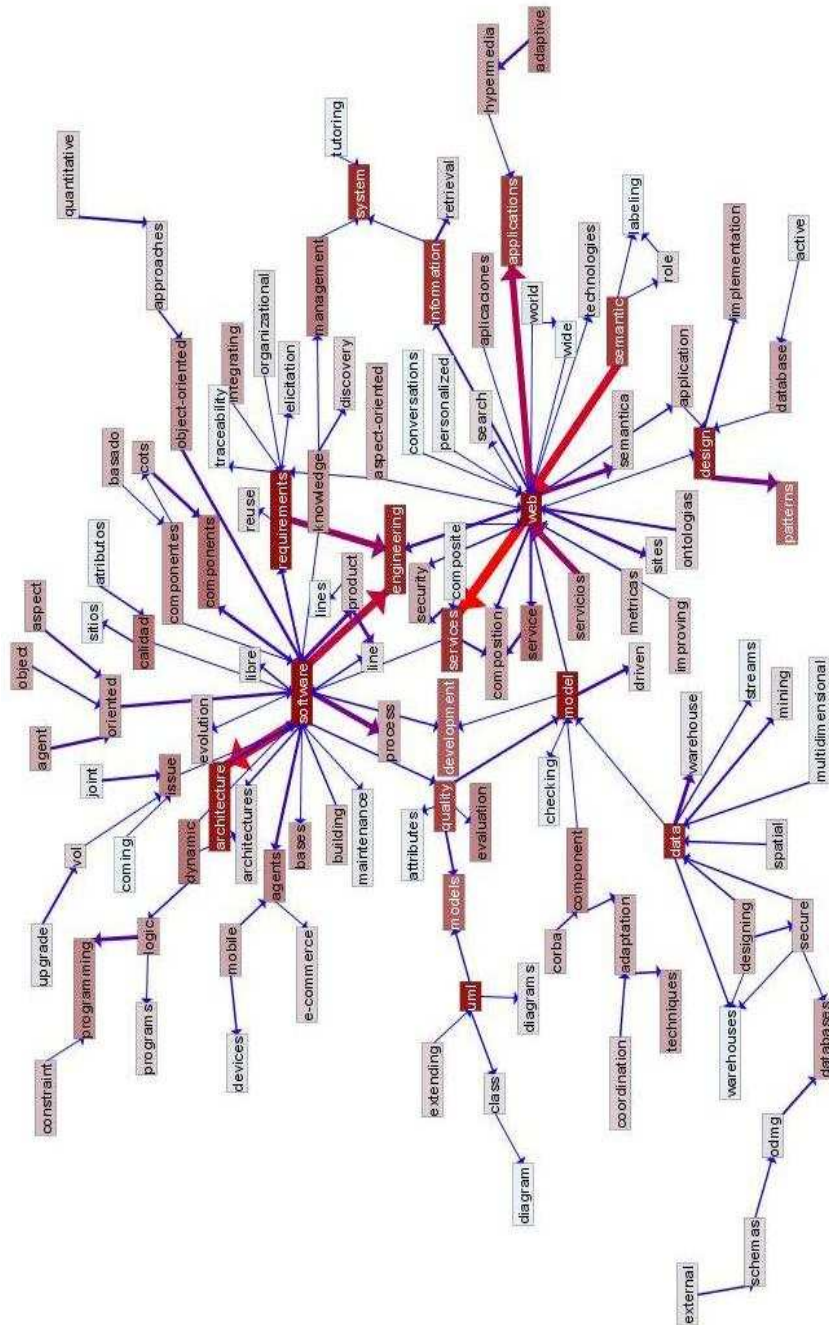


Figura 6: Centros de interés de las JISBD extraídos en los experimentos

Ordenación de Resultados en Consultas Semánticas sobre Bases de Datos

José M. Blázquez, Jesús Arias, Vicente Luque, Luis Sánchez

Universidad Carlos III de Madrid. Dep. de Ingeniería Telemática.
Edificio Torres Quevedo. Avda. de la Universidad, 30. Leganés 28911.
{jmb,jaf,vlc,luis}@it.uc3m.es

Resumen

La integración de datos y aplicaciones siempre ha tenido que hacer frente a los problemas derivados de la heterogeneidad. Muchos han sido los esfuerzos encaminados a resolver este problema lo más automáticamente posible. Estos esfuerzos han recibido un impulso considerable con la irrupción de las tecnologías de Web semántica. Sin embargo, el uso de estas tecnologías no está exento de problemas que deben ser abordados. En concreto, la utilización de ontologías como soporte en determinadas aplicaciones precisa en muchos casos de medidas de similitud semántica entre los conceptos involucrados. En este artículo presentamos una propuesta novedosa con la que hacer frente a esta cuestión.

1. Introducción

Uno de los problemas a los que nos enfrentamos diariamente en la Sociedad de la Información es la cantidad de información disponible. Esto, que en principio es algo positivo puesto que nos permite estar al corriente de temas de interés, se ha convertido en muchos casos en motivo de cierta frustración por el tiempo que hay que dedicar a rastrear entre el conjunto de fuentes de información hasta que se consigue la que se busca, si es que finalmente se llega hasta ella. El caso más conocido es el de la búsqueda de información en la Web.

Pese a los notables avances en los motores de búsqueda, siempre hay situaciones en las que no queda más remedio que inspeccionar un número elevado de sitios Web antes de encontrar la información que se precisa. Otro escenario cada vez más extendido es la búsqueda de un servicio que cumpla una serie de requisitos.

El principio fundamental sobre el que gira todo sistema de información que admite consultas por parte de un usuario o de una aplicación es devolver los resultados que coincidan con dicha consulta y, en la medida de lo posible, presentarlos por orden de relevancia respecto a la consulta formulada. Así, volviendo al ejemplo de las búsquedas en la Web, a partir de una consulta definida como un conjunto de términos, el motor de búsqueda devolverá los sitios Web que considera más relevantes para esos términos.

En este sentido, la irrupción de la Web semántica [2] ha permitido ir un paso más allá y manejar conceptos en vez de simples términos. La definición de conceptos tiene lugar en el marco de las ontologías, que permiten organizarlos jerárquicamente y definir propiedades varias. Dentro del conjunto de relaciones entre conceptos destaca la relación *subClassOf*, que facilita la construcción de la jerarquía. La presencia de esta relación permite que una consulta que solicita documentos anotados con determinado concepto c_1 tenga entre sus resultados documentos en los que no aparece c_1 pero sí aparece c_2 , siendo c_2 subclase de c_1 .

El uso de ontologías en las búsquedas da lugar a una nueva variable a tener en cuenta a la hora de ordenar los resultados devueltos: la similitud semántica entre el conjunto de conceptos de la consulta formulada por el usuario y el conjunto de conceptos usados para anotar cada documento. Esta variable, que puede complicar el problema en el caso de búsquedas de documentos en la Web, es de gran utilidad en otros campos de aplicación en los que la ordenación de resultados tiene menos información para decidir que el resultado A es más o menos relevante que el resultado B . Uno de estos campos, que fue el que motivó este trabajo, es el de las búsquedas semánticas en bases de datos de servicios Web para su integración en *workflows* de servicios. Frente a la ordenación de documentos Web, en los que se puede aplicar técnicas como *tf/idf* [1] para medir cuántas veces aparece en cada documento un término de la consulta y cómo de frecuente es ese término en el conjunto de documentos, en bases de datos simplemente hay resultados que cumplen con los requisitos de la consulta, pero cuyo grado de relevancia frente a ésta no se puede cuantificar con el mismo procedimiento. Por este motivo es necesario abordar el problema de otra manera. En el caso de bases de datos que admiten consultas semánticas, es precisamente este factor el que se puede explotar para obtener la ordenación de resultados deseada.

Existen varias modalidades de búsquedas semánticas en bases de datos. La más habitual es aquella en la que los valores de uno o varios atributos de una tabla toman valores de entre los elementos de una o varias ontologías. Cuando se lanza una consulta, en la que también se usan estas ontologías para filtrar el conjunto de resultados a obtener, tiene lugar el proceso de expansión de la consulta. A modo de ejemplo, supongamos que la ontología contiene los conceptos c_1 , c_2 , c_3 y c_4 , entre otros. Supongamos, asimismo, que c_2 y c_3 son subclases de c_1 y que c_4 es subclase de c_3 . Para una consulta sobre una base de datos en la que el atributo a tenga que ser igual a c_1 , la expansión de la consulta, tras obtener los conceptos subclase de c_1 , convertirá la consulta inicial a otra en la que se solicitan los registros en los que a es

igual a c_1 , c_2 , c_3 o c_4 .

Este procedimiento de expansión de la consulta da lugar a resultados que tienen una distancia semántica respecto a la consulta original, lo cual permite establecer un criterio para la ordenación del conjunto de resultados. Bien es cierto que existen trabajos que ordenan resultados según la relevancia de los atributos no presentes en la consulta, como en [3].

Para el cálculo de la similitud semántica entre la consulta y los resultados es preciso tener una medida de similitud entre conceptos de una ontología. En este artículo se presenta la medida que hemos definido a tal efecto. Para ello se hará un recorrido por los trabajos más relevantes que hay en el estado del arte, para posteriormente introducir nuestra propuesta en la sección 3. La evaluación del algoritmo propuesto tiene lugar en la sección 4. Para cerrar el artículo se exponen las conclusiones obtenidas y el trabajo que se está desarrollando como continuación.

2. Estado del arte

El uso de medidas de similitud es un área de investigación con muchos años de experiencia. Ya en 1965 se puede encontrar un trabajo de Rubenstein y Goodenough acerca de la similitud entre términos del idioma inglés en base a la semejanza del contexto en que aparecen [12]. Para validar su propuesta hicieron un experimento en el que dos grupos de personas, de 15 y 36 sujetos respectivamente, evaluaban la similitud entre 65 pares de sustantivos. La evaluación era un valor numérico entre 0.0, para denotar ausencia de semejanza, y 4.0, que indicaba sinonimia. Este experimento sentó las bases de las pruebas a realizar por las sucesivas propuestas que tuvieron lugar en el campo de las medidas de similitud.

No obstante, el experimento estándar con el cual todos los autores posteriores se han comparado es el que realizaron Miller y Charles [10] en relación con su trabajo en WordNet [9]. En dicho experimento seleccionaron 38 estudiantes de grado para estimar la semejanza entre 30 pares de nombres. Dicha semejanza también se expresaba con un valor entre 0.0 y

4.0. A partir de este experimento, se ha podido evaluar cómo de acertadas han sido las distintas propuestas de similitud entre elementos de una taxonomía, que por lo general estaban pensadas para su uso en WordNet. Dicha evaluación ha consistido en medir la correlación entre los resultados obtenidos por el algoritmo propuesto, y los resultados proporcionados por los sujetos del experimento de Miller y Charles.

Entre los algoritmos presentes en el estado del arte, uno de los más destacados y de los pioneros es el creado por Resnik [11], el cual propone que la similitud entre dos conceptos c_1 y c_2 se puede obtener mediante la expresión:

$$sim(c_1, c_2) = \max_{c \in S(c_1, c_2)} (-\log p(c)) \quad (1)$$

En esta ecuación $S(c_1, c_2)$ representa el conjunto de conceptos de los cuales tanto c_1 como c_2 descienden, mientras que $p(c)$ es la probabilidad del concepto c , la cual veremos más adelante cómo se puede estimar. El término concepto hace referencia a lo que en WordNet se conoce como synset, es decir, el conjunto de términos que hacen referencia a una misma idea.

Para estimar la probabilidad de un concepto c Resnik utiliza la frecuencia de aparición de los términos de ese concepto en el Brown Corpus of American English [4].

Otro trabajo que también hace uso de la probabilidad de un concepto es el propuesto por Lin [7], el cual obtiene de forma teórica la expresión para calcular la similitud entre dos conceptos c_1 y c_2 . Dicha expresión es:

$$sim(c_1, c_2) = \frac{2 * \log p(c_3)}{\log p(c_1) + \log p(c_2)} \quad (2)$$

en donde c_3 es el nodo padre común a c_1 y c_2 más próximo. Un caso particular de esta ecuación es la medida que proponen Wu y Palmer, para quienes la similitud entre c_1 y c_2 se obtiene como:

$$sim(c_1, c_2) = \frac{2 * N_3}{N_1 + N_2 + 2 * N_3} \quad (3)$$

En esta expresión N_1 y N_2 son el número de nodos que hay entre c_1 y c_3 , y entre c_2 y c_3 , respectivamente. N_3 , por su parte, es el número de nodos entre c_3 y el nodo raíz.

Otra medida de similitud a mencionar es la de Leacock y Chodorow [6], quienes consideran que la similitud entre c_1 y c_2 se puede obtener como:

$$sim(c_1, c_2) = -\log \left(\frac{len(c_1, c_2)}{2 * MAX} \right) \quad (4)$$

en donde $len(c_1, c_2)$ es el número de saltos a dar entre c_1 y c_2 , y MAX es el valor máximo de saltos que se pueden dar entre el nodo raíz y los nodos hoja de la taxonomía.

Por último, un trabajo que da unos resultados bastante buenos en WordNet es el de Hirst y St-Onge [5]. En vez de hacer uso de ecuaciones basadas en la probabilidad de los conceptos involucrados, o en contar el número de nodos entre dichos conceptos, lo que hacen es introducir la idea de cadenas léxicas para la obtención de la similitud.

Tras este recorrido por el estado del arte se puede observar la importancia de WordNet en la definición de medidas de similitud, puesto que casi todas las propuestas hasta la fecha van dirigidas a ser usadas con esta taxonomía. No obstante, existen trabajos como [8] en el que se aplican estas medidas de similitud a la taxonomía Gene Ontology¹.

3. Algoritmo

El principal problema que existe entre las medidas de similitud presentadas, y que dio lugar a la creación de un algoritmo alternativo, fue que, como se ha visto en la sección 2, casi todas están pensadas para ser usadas en WordNet. Por este motivo, expresiones como las de las ecuaciones 1 o 2, no pueden ser aplicadas en taxonomías en las que no exista forma de estimar la probabilidad de un concepto c . Por ejemplo, Resnik estima la probabilidad de un concepto mediante la frecuencia de aparición

¹<http://www.geneontology.org>

de los términos asociados a ese concepto dentro de un corpus significativo, como el Brown Corpus of American English. De esta forma, dicha probabilidad se puede estimar aplicando la expresión:

$$\hat{p}(c) = \frac{\text{freq}(c)}{N} \quad (5)$$

en donde $\text{freq}(c)$ es el número de veces que aparecen en el corpus los nombres que corresponden al concepto c o a sus subconceptos, y N es el número total de nombres presentes.

Si se quisiese hacer uso de esta forma de estimar la probabilidad en taxonomías distintas de WordNet, habría que ver cómo medir la frecuencia de aparición de cada concepto. Un posible modo de hacer esto es a partir de las instancias que hay en la base de conocimiento. No obstante, esto acarrea dos problemas. Por un lado, las instancias de una ontología no tienen por qué ser algo estático. Podemos pensar en una ontología para el mundo del periodismo en que, a medida que aparecen nuevos personajes de la vida pública, se van creando nuevas instancias.

Por otro lado, aun en el caso de que tuviésemos una ontología que no va a ser modificada, existe otro problema de carácter conceptual. Veamos para ello un ejemplo: supongamos una ontología para modelar un ambiente universitario. Dicha ontología puede contener la clase *Profesor* para la que se pueden definir las subclases *Catedrático*, *ProfesorTitular*, *Ayudante*, *ProfesorAsociado*, ... Si las instancias de la ontología reflejan el número total de individuos de cada uno de estos estamentos, lo más probable es que el número de instancias de la clase *Catedrático* sea menor que el de la clase *ProfesorTitular*. Si se utilizase la expresión 2 para el cálculo de la similitud semántica, llegaríamos a la conclusión de que el concepto *Catedrático* es menos similar a *Profesor* que el concepto *ProfesorTitular*.

Tras ver estas carencias de las medidas existentes se decidió proponer una alternativa. Dado que el problema que motivó su aparición fue la medida de la similitud en el contexto de la expansión de consultas, en donde la comparación tiene lugar entre un concepto y uno

de sus subconceptos, toda la explicación va a tener lugar suponiendo dos conceptos c_0 y c_1 , en donde c_1 es subclase de c_0 . Esto no implica que c_1 sea un hijo directo de c_0 en la ontología, debido a la transitividad de la propiedad *subClassOf*. Posteriormente se ampliará la discusión al caso general de dos conceptos c_1 y c_2 que no son subclase uno del otro. Con estas premisas en mente, la medida de similitud que se propone se puede expresar como:

$$\text{ssim}(c_0, c_1) = \frac{k * N_0}{k * N_0 + \log(\frac{E_0}{E_1})} \quad (6)$$

en donde N_0 es el número de nodos que hay entre el nodo raíz y c_0 , lo cual permite introducir la noción de que descender un nivel en la ontología supone mayor diferencia semántica en la parte alta de la ontología (la de los conceptos más genéricos) que en la parte baja (donde están los conceptos más específicos).

El cociente $\frac{E_0}{E_1}$ representa la relación entre la información que aporta c_0 y la información que aporta c_1 . Si suponemos que el nodo raíz contiene el 100 % de la información presente en la ontología, cada uno de los nodos inferiores contendrá una porción de esa información. La forma en que se modela el reparto de dicha información está relacionada con la estructura de la ontología. Si una clase c tiene una cantidad de información E , lo que vamos a suponer es que cada una de sus n subclases tendrá una cantidad de información $\frac{E}{n}$, es decir, que la información se va repartiendo equitativamente entre las clases hijas. Con esto lo que se expresa es que, cuanto menor es el número de clases en que se divide una clase, mayor es el parecido entre estas subclases y la clase padre. El motivo de esto es bien claro. Las taxonomías en general, y las ontologías en particular, se diseñan con el objetivo de organizar la información de forma manejable. Para ello, cuando una clase tiene un número muy elevado de subclases, se suele crear clases intermedias que permitan definir grupos de tamaño razonable. La agrupación de las subclases en clases intermedias se hace a partir de características cualitativas comunes. Por ello, cuanto más específicas sean estas características, más especializada estará la clase intermedia y más semejantes serán

respecto a ella las subclases que pasen a ser sus hijas. Adicionalmente también se cumplirá que cuanto más específica sea la clase intermedia, habrá un menor número de subclases que puedan a pasar a ser hijas suyas. Es por esto que podemos asumir que existe relación directa entre el número de subclases de una clase y la similitud semántica entre la clase y sus subclases.

Retomando la expresión 6, cabe destacar que para el cálculo de $\frac{E_0}{E_1}$ no hace falta obtener E_0 y E_1 por separado en relación al nodo raíz, puesto que $E_0 = r * E_1$, en donde r se puede obtener recorriendo la ontología desde E_0 hasta E_1 . También es importante señalar que el cálculo de la similitud se hace teniendo en cuenta que c_0 es superclase de c_1 , de modo que la tasa de información siempre se ha de obtener como $\frac{E_0}{E_1}$, y no como $\frac{E_1}{E_0}$.

El último elemento por determinar en la ecuación 6 es el parámetro k . Este parámetro, como se puede observar, permite ajustar el peso que se da a las dos variables que se manejan en la fórmula: profundidad de c_0 y tasa de información. Su obtención ha de ser obtenida empíricamente y por ello los primeros experimentos que se realizaron en torno al algoritmo que se presenta fueron destinados a este fin.

Una vez explicado cómo calcular la similitud entre un concepto c_0 y uno de sus subconceptos c_1 , lo que queda por hacer es ver cómo extender esto al caso de dos conceptos c_1 y c_2 que no son subclase uno del otro. Para ello es preciso introducir el concepto de distancia. La medida propuesta en la expresión 6 da lugar a un valor entre 0.0 (ninguna similitud) y 1.0 (mismo concepto). De este modo, la distancia $dist$ entre dos conceptos se puede definir a partir de su similitud semántica $ssim$ como:

$$dist(c_1, c_2) = \frac{1}{ssim(c_1, c_2)} - 1 \quad (7)$$

Cuando c_1 y c_2 sean el mismo concepto, $ssim$ valdrá 1 y la distancia entre ambos será 0. Análogamente, cuando $ssim$ valga 0, la distancia entre ambos será infinito. Con esto en mente podemos calcular la similitud entre c_1 y c_2 a partir de la distancia entre ambos.

Si C es el conjunto de conceptos que son superclase de c_1 y c_2 , esta distancia la podemos expresar como:

$$dist(c_1, c_2) = \min_{c_0 \in C} (dist(c_0, c_1) + dist(c_0, c_2)) \quad (8)$$

Haciendo las sustituciones pertinentes según la ecuación 7, se llega a que $ssim(c_1, c_2)$ vale:

$$s_{12} = \max_{c_0 \in C} \frac{s_{01} * s_{02}}{s_{01} + s_{02} - s_{01} * s_{02}} \quad (9)$$

En esta ecuación, s_{ij} representa lo que en el resto del artículo se ha denominado $ssim(c_i, c_j)$, que ha sido sustituido por cuestión de espacio. Como se puede observar, la ecuación 6 es un caso particular de ésta, en el que $s_{02} = 1$.

4. Experimentos

La evaluación de la medida de similitud propuesta se hizo repitiendo el experimento de Miller y Charles, explicado en la sección 2, y calculando la correlación entre los resultados obtenidos en ambos experimentos. La versión utilizada fue WordNet 1.7.1. Dado que el algoritmo expuesto está pensado para medidas de similitud en ontologías, fue preciso hacer una traducción previa de WordNet a OWL. En dicha traducción se usó simplemente el fichero de nombres, puesto que el experimento de Miller y Charles está hecho precisamente sobre nombres. Además, no se tradujeron todas las relaciones que incluye WordNet entre conceptos, sino sólo aquellas que dan lugar a la taxonomía. Concretamente, se llevaron a cabo las siguientes traducciones:

```
a hyponymOf b -> a subClassOf b
a hypernymOf b -> b subClassOf a
```

El conjunto de triplas generadas se almacenó en una base de datos MySQL. El manejo de las triplas se hizo con Jena².

El primer objetivo de los experimentos fue obtener el comportamiento del algoritmo en función del parámetro k . Para ello se varió entre 0.25 y 2.5 en saltos de 0.25. En el cuadro 1

²<http://jena.sourceforge.net>

k	Correlación
0.25	0.7426
0.5	0.7967
0.75	0.8166
1.0	0.8244
1.25	0.8269
1.5	0.8264
1.75	0.8243
2.0	0.8213
2.25	0.8176
2.5	0.8135

Cuadro 1: Variación del parámetro k.

se puede ver la correlación que se produce en cada caso al compararlo con Miller y Charles.

Una vez determinado el rango de valores más adecuados para k , dentro del contexto de WordNet, el siguiente experimento consistió en comparar los resultados obtenidos con otras medidas disponibles en el estado del arte, también sobre WordNet 1.7.1. En el caso de estas medidas, en vez de programarlas, se utilizó el software provisto por Pedersen³. Concretamente, la versión utilizada fue la 0.16, que nos permitía hacer la evaluación sobre WordNet 1.7.1. En el cuadro 3 se pueden ver los resultados de este experimento. Las correspondientes correlaciones aparecen en el cuadro 2 y en la figura 1. Los resultados de nuestra medida de similitud que se muestran son los correspondientes a $k = 1,25$. También calculamos estas correlaciones con el método *leave-one-out*, en el que se calcula la correlación sobre el conjunto de resultados menos uno, dejando fuera en cada iteración un elemento distinto, y se hace la media sobre todas las correlaciones. Esto permite suavizar el efecto de algún caso en el que la discrepancia entre el experimento original y los experimentos automáticos sea muy elevada debido, por ejemplo, al diseño de la taxonomía. Se puede observar en dicha gráfica que los resultados de correlación mejoran para todos los algoritmos evaluados.

Los resultados obtenidos en estos experimentos avalan la validez de la medida de similitud propuesta en este artículo. De hecho,

Medida	Correlación
Hirst, St-Onge	0.6853
Leacock, Chodorow	0.8208
Lin	0.7535
Pedersen, Banerjee	0.7635
Resnik	0.7923
Wu, Palmer	0.7792
ssim	0.8269

Cuadro 2: Correlaciones con Miller y Charles.

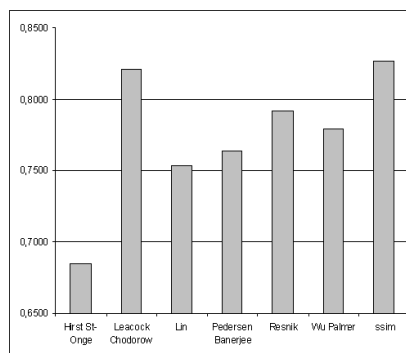


Figura 1: Correlaciones con Miller y Charles.

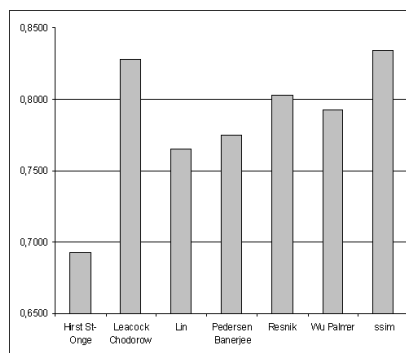


Figura 2: Correlaciones leave-one-out con Miller y Charles.

³<http://www.d.umn.edu/~tpederse/similarity.html>

Parejas de nombres	CM	HS	LC	L	PB	R	WP	ssim
car-automobile	3.92	16	3.5835	1	1	6.0846	1	1
gem-jewel	3.84	16	3.5835	1	1	10.4944	1	1
journey-voyage	3.84	4	2.8904	0.8231	0.5	7.7122	0.9333	0.7397
boy-lad	3.76	5	2.8904	0.8156	0.5	7.5499	0.9231	0.7917
coast-shore	3.70	4	2.8904	0.9664	0.5	8.9103	0.9231	0.8505
asylum-madhouse	3.61	4	2.8904	0.9826	0.5	11.4752	0.9474	1
magician-wizard	3.50	16	3.5835	1	1	10.9644	1	1
midday-noon	3.42	16	3.5835	1	1	10.3766	1	1
furnace-stove	3.11	5	1.5041	0.2169	0.125	2.5158	0.5333	0.1883
food-fruit	3.08	0	1.5041	0.1245	0.125	0.8287	0.3636	0.0541
bird-cock	3.05	6	2.8904	0.804	0.5	7.714	0.9474	0.7543
bird-crane	2.97	5	2.1972	0	0.25	7.714	0.8571	0.5631
tool-implement	2.95	4	2.8904	0.9203	0.5	7.0726	0.9231	0.6924
brother-monk	2.82	4	2.8904	0.2524	0.5	10.9644	0.9333	0.2406
lad-brother	1.66	3	1.9741	0.2875	0.2	2.5144	0.6667	0.2742
crane-implement	1.68	3	1.9741	0.3186	0.2	3.7236	0.7143	0.3571
journey-car	1.16	0	0.9445	0	0.0714	0	0.1333	0
monk-oracle	1.10	0	1.5041	0.2241	0.125	2.5144	0.5333	0.2285
cemetery-woodland	0.95	0	1.2809	0.0774	0.1	0.8287	0.3077	0.0415
food-rooster	0.89	0	0.9445	0.0983	0.0714	0.8287	0.2353	0.043
coast-hill	0.87	3	1.9741	0.7089	0.2	6.5455	0.7143	0.3472
forest-graveyard	0.84	0	1.2809	0.0774	0.1	0.8287	0.3077	0.0415
shore-woodland	0.63	2	1.7918	0.1358	0.1667	1.3374	0.5455	0.1373
monk-slave	0.55	3	1.9741	0.2448	0.2	2.5144	0.6667	0.2694
coast-forest	0.42	0	1.6376	0.1317	0.1429	1.3374	0.5	0.1295
lad-wizard	0.42	3	1.9741	0.2701	0.2	2.5144	0.6667	0.292
chord-smile	0.13	0	1.1856	0.2742	0.125	2.7986	0.5	0.1691
glass-magician	0.11	0	1.5041	0.1288	0.125	1.3374	0.4	0.0809
rooster-voyage	0.08	0	0.5878	0	0.05	0	0.0952	0
noon-string	0.08	0	1.0986	0	0.0833	0	0.1538	0

Cuadro 3: Valores de similitud para parejas de nombres, con CM=Charles, Miller; HS=Hirst, St-Onge; LC=Leacock, Chodorow; L=Lin; PB=Pedersen, Banerjee; R=Resnik; WP=Wu, Palmer; ssim=Medida propuesta.

se puede observar que se obtiene el valor más alto de correlación con el experimento de Miller y Charles. No obstante, esto no es lo más importante de estas medidas, puesto que algunas de las usadas para comparar pueden obtener mejores resultados aplicando un corpus de términos apropiado. Lo verdaderamente relevante es que el algoritmo que se propone como alternativa a los ya existentes, pese a no estar pensado específicamente para WordNet, obtiene resultados tan buenos como el resto de algoritmos, entre los que encontramos algunos que sí fueron diseñados para este lexicón.

5. Conclusiones y trabajo futuro

En el presente artículo se ha presentado una medida de similitud semántica entre elementos de una taxonomía que puede ser de gran ayuda en el proceso de integración automática de datos y aplicaciones en los que, debido a la heterogeneidad existente, se ha hecho uso de ontologías que faciliten el proceso.

La conclusión principal que se extrae de este trabajo es la validez del algoritmo propuesto para el cálculo del grado de similitud entre dos nodos de una taxonomía. Los experimentos llevados a cabo así lo atestiguan. Como ventajas adicionales sobre algoritmos del estado del arte cabe destacar en primer lugar su ámbito de aplicación. Al no hacer ninguna suposición sobre la taxonomía sobre la que se van a hacer los cálculos, se puede usar esta medida en cualquier dominio. Como consecuencia adicional, no precisa de datos externos, como puede ser el uso de un corpus, puesto que los cálculos sólo dependen de la estructura de la ontología.

Estas posibilidades de aplicación en otros dominios son las que abren las puertas de nuevas líneas de trabajo sobre las que seguir avanzando. En primer lugar, es preciso determinar en qué grado hay que modificar el parámetro k cuando se utiliza otra taxonomía. Ante esta situación se plantean dos posibilidades. Si el valor óptimo para k en otras taxonomías es semejante al ya obtenido, se puede fijar un valor concreto y dejar la fórmula cerrada. Si no es así, es preciso determinar qué relación tiene con el tamaño y la estructura de la ontología,

para poder obtener el valor más adecuado en cada caso sin necesidad de calcularlo empíricamente.

La ampliación del ámbito de aplicación que se propone no se limita al uso de otras taxonomías sobre las que realizar otros experimentos, sino también a la utilización del algoritmo presentado en otros contextos. Uno de los más atractivos es la categorización automática de servicios Web. Actualmente estamos trabajando en, a partir de la descripción WSDL de un servicio Web, y con la ayuda de WordNet, encontrar la(s) categoría(s), sobre un conjunto de categorías definido previamente, que mejor encaje(n) con ese servicio.

Desde el punto de vista del algoritmo en sí, pese a que los resultados son esperanzadores, también se plantea la introducción de información adicional que pueda mejorar estos resultados en determinadas circunstancias. Si bien toda la discusión es aplicable a taxonomías, es posible centrarla en ontologías, en las que se dispone de información adicional, como puede ser las propiedades de las clases definidas. Adicionalmente, se puede trasladar el problema a buscar la similitud entre clases de diferentes ontologías. Con esto se entra en el terreno del *mapeo* entre ontologías, que es uno de los problemas relativos a la Web semántica que presenta mayores dificultades, y cuya solución supondría un espaldarazo definitivo a la integración de aplicaciones que utilicen datos heterogéneos, anotados incluso con ontologías distintas.

Agradecimientos

Este trabajo ha sido financiado parcialmente por el Ministerio de Ciencia y Tecnología a partir de los proyectos ITACA (TSI2006-13409-C02-01), ITACA 2 (TSI2007-65393-C02-01) e InCare (TSI2006-13390-C02-01), y por el Grupo de Aplicaciones y Servicios Telemáticos del Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid mediante el proyecto CP06: Anotación y Búsqueda de Recursos Educativos.

Referencias

- [1] Baeza-Yates, R., Ribeiro-Neto, B., *Modern Information Retrieval*, Addison-Wesley, 1999.
- [2] Berners-Lee, T., Hendler, J., Lassila, O., *The Semantic Web*, Scientific American, 284(5):34–43, 2001.
- [3] Chaudhuri, S., Das, G., Hristidis, V., Weikum, G., *Probabilistic information retrieval approach for ranking of database query results*, ACM Press, 31(3):1134–1168, 2006.
- [4] Francis, W., Kucera, H., *Frequency Analysis of English Usage*, Houghton Mifflin Co., 1982.
- [5] Hirst, G., St-Onge, D., *Lexical chains as representation of context for the detection and correction of malapropisms*, en Fellbaum, C., *WordNet: An Electronic Lexical Database*, MIT Press, 265–283, 1998.
- [6] Leacock, C., Chodorow, M., *Combining local context and WordNet similarity for word sense identification*, en Fellbaum, C., *WordNet: An Electronic Lexical Database*, MIT Press, 265–283, 1998.
- [7] Lin, D., *An information-theoretic definition of similarity*, Proc 15th International Conference on Machine Learning, 296–304, 1998.
- [8] Lord, P.W., Stevens, R.D., Brass, A., Goble, C.A., *Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation*, Bioinformatics, 19(10), 1275–1283, 2003.
- [9] Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K., *WordNet: an on-line lexical database*, International Journal of Lexicography, 3(4), 1990.
- [10] Miller, G.A., Charles, W.G., *Contextual Correlates of Semantic Similarity*, Language and Cognitive Processes, 6(1):1–28, 1991.
- [11] Resnik, P., *Selection and Information: A Class-Based Approach to Lexical Relationships*, Ph.D Dissertation, 1993.
- [12] Rubenstein, H., Goodenough, J.B., *Contextual Correlates of Synonymy*, Communications of the ACM, 8(10):627–633, 1965.

An Advanced Query Capability Description Framework Implementation

Carlos Rivero⁽¹⁾, David Ruiz⁽²⁾

⁽¹⁾ TB Solutions.

Cl. Carlos de Cepeda, 5. Sevilla 41005.
carlos.rivero@tb-solutions.com

⁽²⁾ Universidad de Sevilla. Dep. de Lenguajes y Sistemas Informáticos.
ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012.
druiz@us.es

Abstract

The Web is the largest knowledge repository of the Humanity, and the reason for its great success is that the information it stores is available in a friendly format by using friendly interaction mechanisms. Software agents may access this repository, which is paving the way for new opportunities to integrate current applications and information. Agents must simulate a person interacting with search forms and extracting information from the resulting pages by means of an API. Using such APIs may lead to serious maintenance problems; thus, it is not surprising that some companies have developed commercial tools by means of which data islands may be queried by using languages such as SQL. In this paper, we focus on the Advanced Query Capability Description Framework which is able to query search forms in a SQL way. We propose an implementation and recap in its main drawbacks.

1. Introduction

The Web Services or the Semantic Web initiatives [1, 2] provide excellent technologies to ease creating new data islands ready to feed software agents, but the current Web's value comes from the integration of existing data islands [4, 5, 6, 7, 9, 10]. That is, they need to be

reengineered, which is difficult if there is not a clear separation between their business and presentation layers. In such cases, the only feasible solution may be to design an enquiring system that reuses the existing search forms and navigates through the resulting pages by using an API that attempts to emulate the behaviour of a human being [8].

It is thus not surprising that some companies have specialised in designing tools to alleviate these problems. Denode (denodo.com), Fetch (fetch.com), or Kapow (kapowtech.com) range amongst the most successful such companies. Despite these tools are very valuable, we have found several drawbacks, namely:

1. They assume that each page provides information about just one real-world entity, but real-world pages tend to provide information about several entities.
2. They rely on SQL, which is very adequate for closed, highly structured data sources but not appropriate for web data sources.
3. They require the fields in the search forms to be closely related to the constraints in the SQL queries, which limits their expressiveness and prevents the data islands and the queries from evolving independently.
4. They do not provide any tools to check for inconsistencies, which are quite frequent

when the data to be integrated comes from distant sources.

5. They just provide a finite-automaton-based tool to define navigation paths and a pattern-based tool to extract information from web pages, but it is generally agreed that one-size-fits-all solutions are not generally applicable.

Their proposals are quite similar, that is:

Denode: it relies on two products, namely: ITDataPort, and ITPilot. The former is an EII solution to integrate SQL data schemes by means of relational operators; it also allows to decompose a query on an integrated scheme into subqueries to the subordinate schemes. ITPilot is a tool that helps extract information from friendly data islands by using a subset of SQL, which is closely related to this research project.

iSoCo: it provides a commercial tool that is closely related to the WebKB project [3]. Its goal is to transform the documents in a data island into data that is structured according to ontology; these results are stored in a database that may be later queried by using the most appropriate language. Therefore, it is kind of a "batch" proposal since its goal is not to retrieve information on demand, but the preprocess makes it available off-line.

Denode's authors propose a model for querying search form in [11] which permits:

- Modelling search forms make more easily working with them.
- Querying search forms with SQL-like syntax queries.
- Computing queries feasibility instead of trying to execute them to know which query would be satisfied by the search form, avoiding a *trial-and-error* process.
- Querying multiple search forms and computing their feasibility.

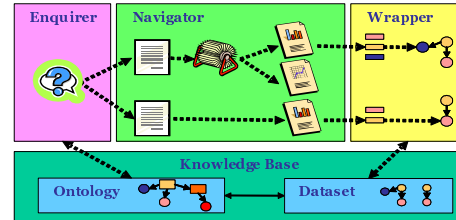


Figure 1: Scheme of the infrastructure needed to build a data island.

Making a query over a search form implies fill it with the query conditions. On the other hand, computing queries feasibility without executing them avoids users to write queries by a *trial-and-error* process. This paper describes an implementation of this framework and analyses it working with search forms. It is organised as follows: in Section 2, we present a summary of the work presented in [11]; Section 3 describes our framework implementation; in Section 4, we conclude and discuss about the solutions we have adopted in some situations.

2. The Advanced Query Capability Framework

2.1. Data islands

Figure 1 depicts the modules necessary to develop a data island. The most important is KnowledgeBase since it helps model the structure of the information a data island provides, but also acts as a cache in which the results of the queries is stored temporarily. Obviously, it uses W3C standards such as OWL, RDF and SPARQL to describe ontologies, represent data or design queries, respectively [2, 4, 12]. This module simplifies it all; the problem is how to feed its dataset, which is the goal of the Enquirer, the Navigator, and the Wrapper.

The Enquirer gets SQL-like queries and transforms them into filled forms. This problem is not trivial at all since, in general, the constraints in a query need not be closely related to the search fields in the corresponding form. Thus, the Enquirer must try to

make the most of the search forms, but some queries may also require to use the existing dataset. The key is to try to make the queries independent from the search forms, since this helps construct robust software agents that may evolve independently from the data islands from which they retrieve the data they require. This paper focus on an implementation of this module using the model proposed by Pan *et al* [11].

The **Navigator** cares of executing the forms provided by the **Enquirer** and navigates through the results to fetch data pages. Neither is this task trivial, since a form may lead to a no-results page, to a summary page, or to a data page with information in which we are not interested (this is usual in virtual shops, since they usually provide information about related products with big savings, for instance). As a conclusion, the **Navigator** must be able to navigate intelligently through the results provided by a search form.

The **Wrapper** is responsible for analysing the data pages provided by the **Navigator** and extracting the data in which we are interested; furthermore, it must structure it according to a given ontology. Needless to say, our framework must provide several information extraction techniques, so that engineers may choose the one that best works with the data islands they need to integrate.

2.2. Relation Model

Each search form exports a combination of base relations. Each base relation is composed of attributes and search-methods, comprised of three elements:

Mandatory capabilities: constraints that must be satisfied by the queries sent to the relation through this particular search-method¹.

Permitted capabilities: query capabilities offered by the relation, once the negative capabilities has been satisfied².

¹In the original paper, this set is referred to as negative capabilities

²In the original paper, positive capabilities

Figure 2: Amazon DVDs search form.

Output: set of attributes which appear in the response of the queries sent against the relation.

Both mandatory and permitted capabilities are represented through the 4-tuple (attribute, operator, multiplicity, possible values). For instance, Amazon DVDs search form is shown in Figure 2. The **TITLE** attribute is a mandatory field. Queries by title and actor are searched by keyword (operator **Contains**). An arbitrary number of words can be used to fill in the boxes and the source will perform the logical **AND** operation between them. We model this source as the relation shown in Figure 3.

Not all the relations can be modelled with only one search-method. When a relation presents mutually excluding options, more than a search-method are needed.

2.3. Query Model

A base query is a query which involves only one relation, for instance:

```
select *
from AmazonDVDs
where QCond1 and QCond2 and ... and QCondN
```

2.4. Matching queries with search-methods

In order to execute a base query against a relation through a certain search-method, there are two simple required conditions not presented for the shake of simplicity.

```

AmazonDVDs=
{
  TITLE:String,
  ACTOR: String,
  PRICE: Enumerated(Any,
                    $10 & Under,
                    $15 & Under,
                    $20 & Under),
  FORMAT: Enumerated(Any,
                     IMAX,
                     Full Screen,
                     Unmatted),
}
{
  NEGATIVE {
    (TITLE, Contains, 1 ANY) }
  POSITIVE {
    (TITLE, Contains, +, ANY)
    (ACTOR, Contains, +, ANY)
    (PRICE, =, 1,
     {'Any',
      '$10 & Under',
      '$15 & Under',
      '$20 & Under'})
    (FORMAT, =, 1,
     {'Any',
      'IMAX',
      'Full Screen',
      'Unmatted'})}
  OUTPUT {
    TITLE, ACTOR, PRICE, FORMAT}
}

```

Figure 3: Relation model for the Amazon DVDs example.

2.5. Post-processing techniques

Query conditions involving those attributes present in the output set can always be performed by the mediator, by applying post-processing techniques.

Another post-processing technique consists on compute which operator can be emulated by another operator and applying the more general one. An operator can be contained into another.

In query execution, it must be difference between the native and the extended capabilities, since queries could be less efficient. So, it must use post-processing only when needed. Since the original work establishes that users decide if post-processing have to be applied or not.

2.6. Global schema relations

Once the base relations have been created, each relation of the global schema is defined by a query involving the base relations, in a similar way to the definition of views in a conventional database.

A view is defined through a relational algebra expression involving relations. The algebraic operators permitted are: union, join, selection, projection, order by, and aggregation.

2.7. Query execution

Once the global relations have been created and their capabilities generated, the mediator is able to accept queries over them.

3. Implementation

3.1. Architecture

The architecture we have adopted to build a framework with these requirements is shown in Figure 4. Query Capability Computer takes the incoming relations and, for global schema relations, computes their global query capabilities. Query Executor receives the obtained relations and some queries. It sends both inputs to the Matcher. Matcher processes each query and relation, deciding if the query over that relation is feasible or not.

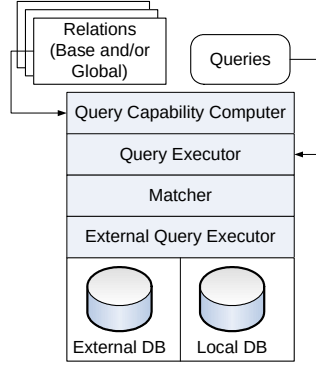


Figure 4: Implementation Architecture.

If it is feasible, External Query Executor executes the query over the relation. Execution over real search forms is simulated by External DB, returned data is stored in Local DB. If post-processing techniques are active, local data is filtered to get the final information. Figure 5 shows an Activity Diagram which explains the system behaviour.

3.2. Static View

The resulting analysis class diagram of the framework is shown in figure 6. Relation has Attributes and SearchMethods. SearchMethod contains two sets of FourUple which correspond to mandatory and permitted sets and a couple of attributes to store its name and the resulting output. FourUple is comprised of attribute name, operator, multiplicity and a list of possible values. FourUple is responsible of compute if another FourUple is permitted, that is, $A \supseteq B$, being both A and B 4-tuple. A less restrictive 4-tuple is of the form:

(A, ANY, +, ANY)

Being X a 4-tuple and QCond a query condition, QCond is satisfied by X if:

$X = (A1, OP1, MUL1, POSS_VALUES1)$
 $QCond = (A2 OP2 VAL)$

Then:

$A1 = A2$

$OP1 = OP2$ or $OP2$ contained in $OP1$
 $VAL \in POSS_VALUES1$

GlobalSchemaRelation extends Relation and contains one OperationNode. UnionNode implements OperationNode and has two OperatorNodes, left and right nodes. Each node can be a GlobalSchemaRelation or a BaseQuery, which holds a QueryCondition list. Each base query condition forms a Boolean expression joining them with AND operator. Thanks to this indirection, we have a recursive structure and we can handle complex relations and queries.

3.3. Design

Figure 7 shows the design diagram class. User can configure the system by indicating if post-processing techniques must be used or not, even which one of the two different techniques we have seen. Configuration is the class responsible of this. Now, QCQueryExecutorFacade executes a base query or a global schema query. QCQueryExecutorImpl uses PrePostProcessing for the post-processing management and has to ask to QueriesSearchMethodsMatcher if the query is feasible or not. If it is feasible, then ExternalQueryExecutor takes care of executing the query over the external or local database and copies data into local. In this case, SQLQueryExecutor is the responsible of query execution since we have use a SQL-DB to simulate real search forms.

4. Conclusions

In this paper, we present an implementation of the framework described in [11]. We conclude that it is useful in order to represent a wide variety of search forms, and simple since a human can understand the form model because it is not a complex representation.

At this time, we have implemented a 30-40 %: base relations, base queries, matching, post-processing, global schema relations and union operator. All the problems we have encountered are not answered directly in the

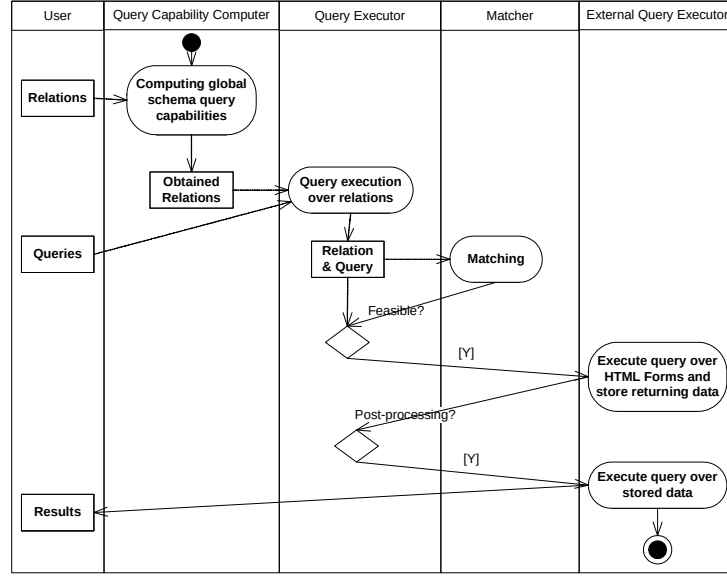


Figure 5: Activity Diagram.

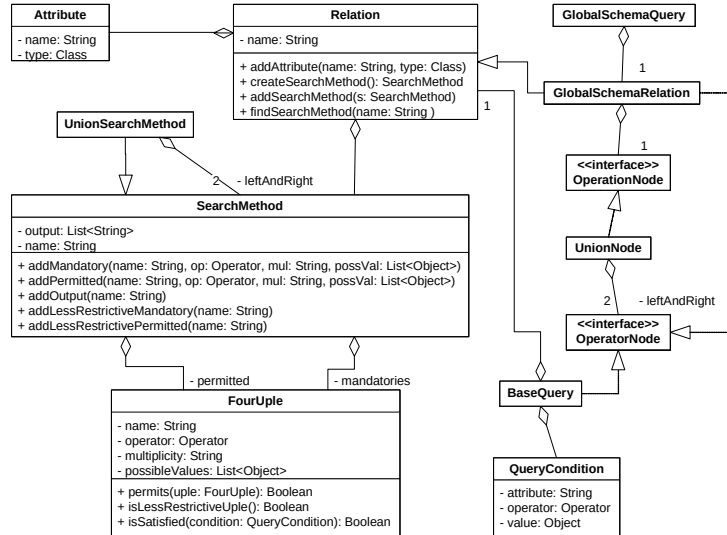


Figure 6: Analysis Static View.

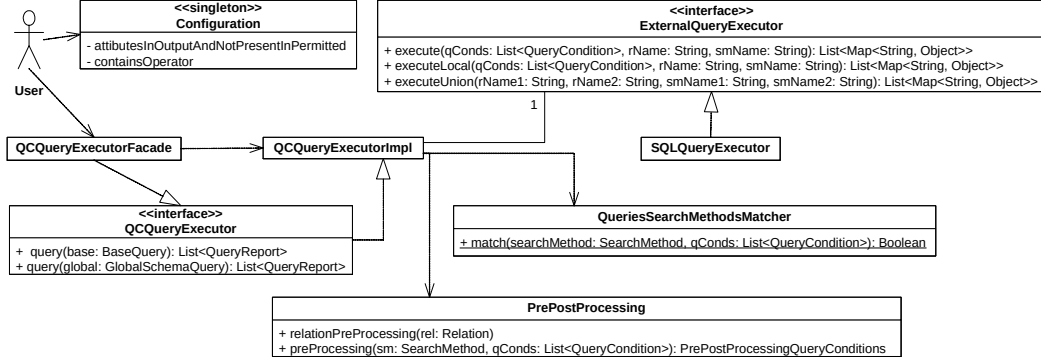


Figure 7: Design Static View.

original work but the framework is flexible enough to solve them. That is:

More than one search-methods [11] mentions that some search forms have to be modelled using more than one search-method but does not mention how deal with such kind of relations.

We have added a search-method name to know which method is using the framework to execute a query. If a query over a search-method is feasible, system keeps execution query results for further queries.

Fields querying complex dataset There are search form fields that query a complex dataset, for instance, Amazon DVDs offers a searching form where we can looking for movies in which it plays a concrete actor/actress, then, returns films containing an actor list each film.

How can we manage these situations? We have decided, in cases it should be applied, take those fields like a character string with actors/actresses separated by a token and query them using Contains operator.

Input field datatype different of output field datatype Other case is a field of a datatype that can be queried and the search form offers another type.

For instance, Amazon DVDs permits querying them by a price range, and offers the exact DVD price. In this context, we have an enumerated field which indicates a price range (PRICE attribute), there is a 4-tuple permitting query it in the permitted set. There is another attribute in output indicating the numeric price (REAL_PRICE attribute). In pre-processing, system removes price query condition (the enumerated one), executes the rest of the query and, by post-processing, filters out the DVDs which price is not in the indicated range.

Negative and disjunctive conditions When writing complex queries, we can use other Boolean operators rather than AND.

The search form can permit NOT operator, in that case, we can query attributes present in mandatory and/or permitted sets. If the search form does not permit it, we can only query permitted attributes using them as filters in post-processing techniques, mandatory attributes can never be queried in this case. OR operator is always permitted, to ease the work we will adopt the Normal Conjunctive Form.

Fields enabling/disabling other fields Some search forms have fields that enable/disable other fields.

To solve this problem we have as search-methods as different field combinations the

search form can offer. This approach can be very inefficient because of the high number of queries the system may have to handle.

Comparing operators using containment

After explaining the operator containment, there are some parts of the original work describing the usage of “ $\supseteq$ ” operator to comparing two operators. For instance, after compute mandatory tuples set in union, we can have redundant restrictions, so less restrictive tuples can be removed:

$X = (A, OP1, MUL1, VALUES1)$
 $Y = (A, OP2, MUL2, VALUES2)$

X is less restrictive if:

$OP1 = OP2$
 $MUL1 \geq MUL2$
 ...

We think that instead of “ $\supseteq$ ” using the contains operator comparison improve the final results.

Acknowledgements

We are thankful to Dr. Rafael Corchuelo for his insightful comments.

References

- [1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer Verlag, 2004.
- [2] G. Antoniou and F. Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.
- [3] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artif. Intell.*, 118(1-2):69–113, 2000.
- [4] J. Davies, M. Lytras, and A. Shteth. Semantic-based knowledge management. *IEEE Internet Computing*, page In press, 2007.
- [5] A. Doan, N. Noy, and A.Y. Halevy. Introduction to the special issue on semantic integration. *SIGMOD Record*, 33(4):11–13, 2004.
- [6] B. Gold-Bernstein and W. Ruh. *Enterprise Integration*. 2004.
- [7] S. Kambhampati and C.A. Knoblock. Information integration on the web. *IEEE Intelligent Systems*, 18(5):14–15, 2003.
- [8] M. Lippert and S. Roock. *Refactoring in Large Software Projects*. John Wiley & Sons, 2006.
- [9] N.F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.
- [10] N.F. Noy, A. Doan, and A.Y. Halev. Semantic integration. *AI Magazine*, 2005.
- [11] A. Pan, P. Montoto, A. Molano, M. Álvarez, J. Raposo, and A. Viña. A model for advanced query capability description in mediator systems. In *ICEIS*, pages 140–147, 2002.
- [12] J. Saarela. *The SPARQL Handbook*. Morgan Kaufmann, 2007.

Acelerando el Desarrollo de Extractores de Información con SRV

Pablo Palacios, José L. Arjona, José L. Álvarez

Universidad de Huelva. Dep. de Tecnologías de la Información.
Escuela Politécnica Superior. Ctra. Huelva-La Rábida. Palos de la Frontera 21071.
{pablo.palacios,jose.arjona,alvarez}@diesia.uhu.es

Resumen

En los últimos años Internet se ha ido expandiendo y convirtiendo de facto en la principal fuente de información en la actualidad. Es por tanto, muy interesante poder acceder a ella de forma automática. A lo largo de los años han surgido diferentes técnicas que nos permiten construir de forma automática extractores de información. Una de las más prometedoras han sido las técnicas inductivas. Dentro de las cuales, SRV ha demostrado tener un gran potencial, sin embargo adolece de un problema común a las técnicas inductivas: un entrenamiento poco eficiente. En este artículo presentaremos una línea de trabajo que trata de hallar mecanismos que nos permitan mejorar el proceso de entrenamiento y acelerar el desarrollo de extractores de información.

1. Introducción

La Web se ha convertido en la principal fuente de conocimiento en muchas áreas. Desafortunadamente, aunque es posible acceder a las casi infinitas webs que existen desde prácticamente cualquier ordenador, actualmente sólo un ser humano es capaz de obtener de ellas información útil. Durante los últimos años han ido surgiendo diferentes alternativas que tratan de obtener de forma automática extractores de información. Este tipo de alternativas están basadas en el aprendizaje automático que se pueden clasificar en dos grandes grupos el no supervisado, y el supervisado.

La extracción no supervisada parte de un conjunto de ejemplos de los que a priori no

sabemos nada y trata de encontrar características comunes entre ellos para agruparlos en clases. Su principal ventaja es la rapidez con la que son capaces de extraer esas características pero no son capaces de trabajar en páginas webs cuya estructura no es fija y puede adoptar diferentes formas para ejemplos de la misma clase. Ejemplos de estos algoritmos son: RoadRunner [1], DEPTA [2] y EXALGA [3].

Los algoritmos supervisados parten de un conjunto de ejemplos que han sido previamente etiquetados y agrupados y tratan de identificar las características comunes de esos grupos ejemplos. Aunque este tipo de algoritmos son capaces de funcionar mucho mejor que los no supervisados en entornos con estructura variable, su principal desventaja es que el proceso de aprendizaje es mucho más lento y costoso. Representantes de estos algoritmos son: SRV [4], STALKER [5] y WEIN [6].

En este artículo nos centraremos en uno de los más prometedores, el algoritmo SRV enmarcado dentro del proyecto WebKB. SRV se basa en un algoritmo de aprendizaje relacional conocido como FOIL [7] y parte de un conjunto de ejemplos positivos y negativos definidos como cláusulas de Horn, además aporta un conjunto extra de predicados que definen el espacio de búsqueda. A partir de ellos y mediante una estrategia top-down trata de crear una teoría que permite explicar los ejemplos.

El resto del artículo se organiza de la siguiente manera: la siguiente sección presenta un caso de estudio típico de extracción de conocimiento en la web; la sección 3 ilustra como funciona SRV (FOIL) a la hora de extraer información de una página web; la sección 4 pre-

senta una posible mejora de FOIL de cara a reducir el costo asociado a su entrenamiento; por último la sección 5 presenta trabajos futuros y muestra los siguientes pasos a dar en nuestro camino de mejorar FOIL.

2. Caso de estudio

Como caso de estudio utilizaremos la base de datos sobre publicaciones conocida como DBLP que se encuentra en la url <http://dblp.uni-trier.de/>. Esta base de datos permite obtener información acerca de autores, congresos, publicaciones, revistas, etc. Una página de ejemplo puede verse en la figura 1.

Nuestro objetivo al trabajar sobre DBLP es ser capaz de determinar a partir de la página de un autor información relevante sobre el mismo, como su nombre, títulos de artículos, coautores, congresos y revistas. Por ejemplo, de la figura 1 podríamos obtener las publicaciones realizadas por el correspondiente autor.

3. El algoritmo SRV

El corazón de SRV es el algoritmo FOIL. FOIL es uno de los técnicas de Programación Lógica Inductiva más utilizadas para la extracción de información. SRV ha demostrado además su utilidad práctica. FOIL representa la información como un conjunto de predicados entre los que distingue dos tipos:

Ejemplos: estos predicados representan la información que FOIL debe aprender, pueden ser de dos clases: positivos que indican ejemplos válidos o negativos que indican ejemplos no válidos. El conjunto de ejemplos positivos forma una relación denominada R .

Predicados de soporte: estos predicados representan información/características adicionales que permiten a FOIL aprender los ejemplos. En general suelen representar información acerca de la página web a tratar: formatos de texto, posiciones, etc. Estos predicados son lo que dan lugar a los diferentes literales que usará FOIL.

En la figura 2 se presenta el esquema de FOIL: parte de una relación R formada por los ejemplos positivos y de una posible colección de predicados de soporte ya definidos. Para aprender esta teoría FOIL usa una estrategia divide y vencerás, aprendiendo de forma iterativa una nueva cláusula (conjunto de predicados) y eliminando de R las tuplas (ejemplos) cubiertas por la nueva cláusula hasta que no quede ninguna. Una nueva cláusula se genera mediante una especialización repetitiva, comienza con la cláusula más general y añade literales (predicados ligados a variables) al cuerpo hasta que la cláusula no cubra tuplas que no pertenecen a R .

Los nuevos literales a añadir por FOIL son alguno de los siguientes:

- $Q(X_1, X_2, \dots, X_k)$ y $\text{not}Q(X_1, X_2, \dots, X_k)$, donde Q es una relación y X_i es una variable previamente ligada en la cláusula o una nueva variable. Al menos una de las variables debe estar ligada a la cláusula.
- $X_i = X_j$ o $X_i \neq X_j$, donde X_i y X_j son variables ya existentes y del mismo tipo.
- $X_i = c$ o $X_i \neq c$ donde X_i es una variable ya existente y c es una constante del tipo apropiado.
- $X_i \leq X_j$, $X_i > X_j$, $X_i \leq t$ o $X_i > t$ donde X_i y X_j son variables numéricas existentes y t es un umbral fijado por FOIL.

En la elección del siguiente literal a añadir, FOIL se guía a partir de una estrategia *greedy*. En cada paso FOIL trata de crear una cláusula que cubra alguna de la tuplas de R sin cubrir ninguna de las que no pertenecen. Para ello debe elegir aquel literal de todos los posibles que le permita acercarse más a este objetivo. Para cada literal FOIL calcula su *ganancia* mediante una heurística basada en la ganancia de información que el nuevo literal proporciona:

$$I(n^{\oplus}, n^{\ominus}) = -\log_2 \frac{n^{\oplus}}{n^{\oplus} + n^{\ominus}}$$

donde $n^{\oplus}$ indica el número de tuplas cubierto al añadir el literal a la cláusula que están en R

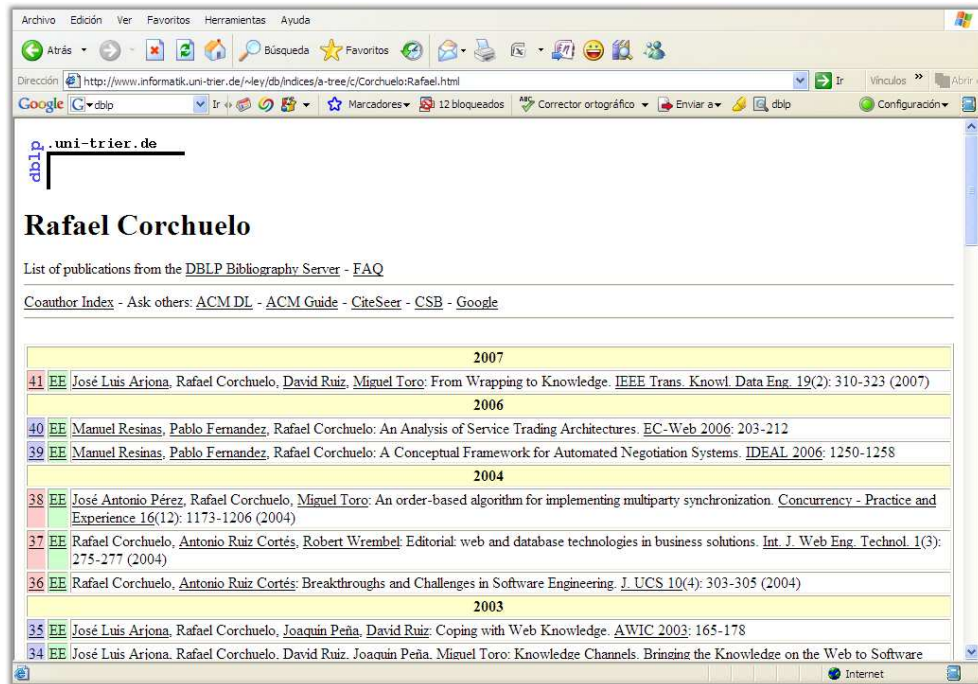


Figura 1: Página web obtenida de DBLP.

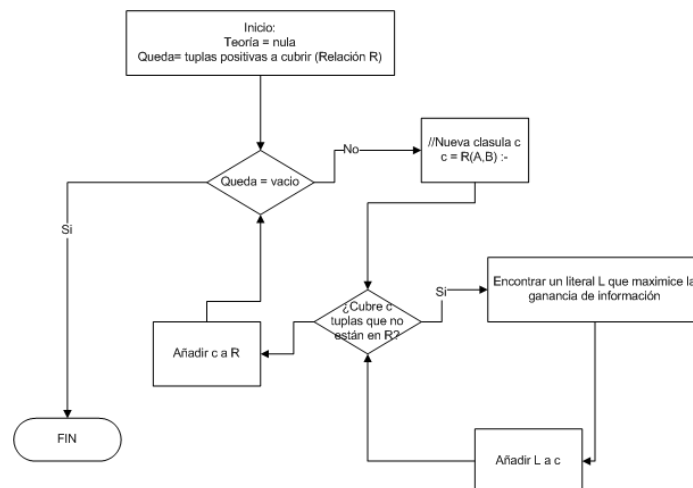


Figura 2: Esquema General de FOIL.

y $n^\ominus$ el número de tuplas cubiertas al añadir el literal a la cláusula que no están en R .

Aunque a priori FOIL es un algoritmo con un gran potencial adolece de un problema importante: la generación de nuevos literales supone un gran costo, por ejemplo supongamos que deseamos calcular cuales son los nuevos literales a añadir para un problema: para un predicado de m variables de la forma $\text{predicado}(x_1, x_2, \dots, x_m)$ es necesario crear al menos $m \times \left(\binom{n}{m} + m! \times n\right)$ donde n es el número de variables existentes en la cláusula. Por ello, para que FOIL puede ser más eficiente es necesario reducir en gran medida este número.

3.1. Usando SRV en DBLP

Para poder trabajar con FOIL sobre DBLP hemos definido una serie de predicados de soporte que detallamos a continuación:

- $hd(x)$: indica que x se encuentra dentro de una etiqueta $\langle h \rangle$.
- $a(x)$: indica que x se encuentra dentro de una etiqueta $\langle a \rangle$.
- $td(x)$: indica que x se encuentra dentro de una celda de una tabla.
- $posterior(x, y)$: indica que x se encuentra después que y en el texto de la página web.
- $anterior(x, y)$: indica que x se encuentra antes que y en el texto de la página web.

Para FOIL esto predicados carecen por completo de significado; por ejemplo, $posterior(x, x)$ es un predicado válido cuando para una persona evidentemente no lo es.

Veamos algunos pasos que sigue FOIL a la hora de determinar como se podría obtener a los coautores de una determinada página web. Para ello utilizaremos un árbol que mostrará como va progresando FOIL a la hora de construir la cláusula final. Este árbol puede observarse en la figura 3.

4. Optimización

Como puede apreciarse en la figura 3 el número de nodos a evaluar por FOIL es elevado y es una de las principales razones de su baja eficiencia. Es necesario, por tanto, reducir el número de nuevos términos que se deben evaluar en cada iteración de FOIL. Una de las formas más simples es ampliar la propia filosofía *greddy* a la creación de los nuevos literales. Nuestra propuesta es reducir el número de nuevos literales a crear y por tanto a evaluar, a un número k . De forma que cuando llegue el momento de decidir el nuevo literal a añadir a la cláusula actual, no crearemos todos los posibles literales sino k de ellos elegidos al azar, de los cuales el de mejor ganancia será el que se añada a la cláusula actual. En la figura 4 puede verse la misma ejecución utilizando esta optimización.

Como puede verse conseguimos así un número mucho menor de nodos a evaluar y una considerable reducción en el cálculo de los nuevos literales. Es posible que la solución que encontremos en este caso pueda ser diferente a la que obtendría FOIL sin la optimización, pero hay que tener en cuenta que para una misma teoría se puede representar a partir de diferentes colecciones de cláusulas y todas ellas son equivalentes.

5. Conclusiones y trabajos futuros

Aunque nuestra optimización parece prometedora sólo es el primer paso de varios para mejorar la eficiencia de FOIL. Los siguientes tratarán de hacer uso de la información proporcionada por la ganancia de información de cada literal con la esperanza de eliminar aquellos literales menos útiles. Otra mejora será tratar de añadir información de contexto para cada predicado: por ejemplo para FOIL los predicados *anterior* y *hd* son iguales pero para un observador humano hay importantes diferencias. En el futuro trataremos de mejorar la riqueza de los predicados de soporte añadiendo información contenida en el árbol DOM de la páginas XHTML así como mediante un preprocesamiento de las páginas webs.

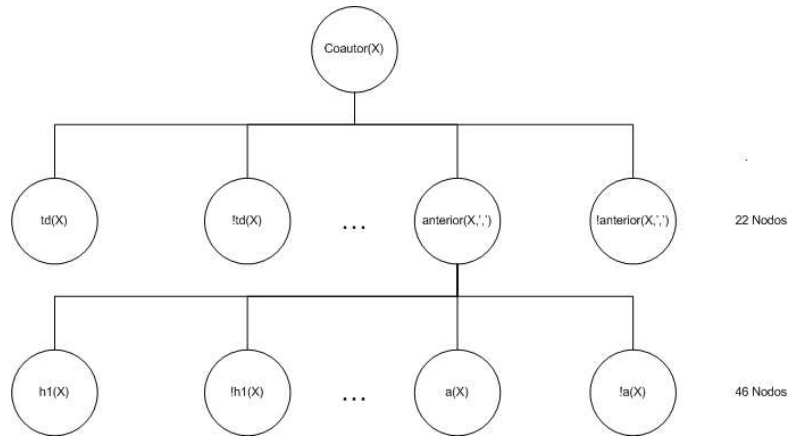


Figura 3: Árbol de ejecución para FOIL.

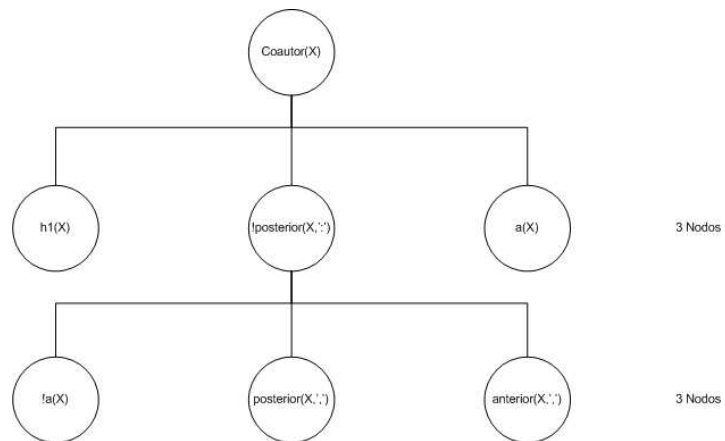


Figura 4: Árbol de ejecución para FOIL optimizado.

Referencias

- [1] Valter Crescenzi and Giansalvatore Mecca and Paolo Merialdo: *RoadRunner: Towards Automatic Data Extraction from Large Web Sites*, Proceedings of 27th International Conference on Very Large Data Bases, (2001), 109-118.
- [2] Yanhong Zhai and Bing Liu: *Web data extraction based on partial tree alignment*, International World Wide Web Conference, (2005), 76 - 85.
- [3] Ling Ma, Nazli Goharian, Abdur Chowdhury and Misun Chung: *Extracting unstructured data from template generated web documents*, Conference on Information and Knowledge Management, (2003), 213 - 215.
- [4] Dayne Freitag: *Information Extraction from HTML: Application of a General Machine Learning Approach*, AAAI, (1998), 517-523.
- [5] Ion Muslea, Steven Minton and Craig A. Knoblock: *Knoblock: Hierarchical Wrapper Induction for Semistructured Information Sources*, Journal of Autonomous Agents and Multi-Agent Systems 4(1/2), (2001), 93 - 114.
- [6] Nicholas Kushmerick: *Wrapper induction: Efficiency and expressiveness*, Artif. Intell. 118, (2000), 15 - 68.
- [7] J. R. Quinlan: *Learning First-Order Definitions of Functions*, Journal of Artificial Intelligence Research, 5, (1996), 139-161.

Uso del Recurso Lingüístico WordNet en la Expansión de Consultas

Francisco J. Pinto

Universidad de la Coruña. Dep. de Ciencias de Computación
Campus de Elviña, s/n. A Coruña 15071
fjoao@udc.es

Resumen

Este trabajo describe la experimentación llevada a cabo para probar la efectividad de la expansión de consultas utilizando una metodología de la evaluación basada en la simulación bajo el modelo del usuario. Realizamos diferentes experimentos de consultas utilizando la información lingüística extraída desde WordNet. Los experimentos muestran que la expansión no es capaz de mejorar la ejecución de la recuperación de información sin la selección del significado correcto de la palabra de entre el conjunto de posibles significados; a este proceso le llamaremos desambiguación del sentido de la palabra.

1. Introducción

La recuperación de datos, en el contexto de un sistema de IR, consiste básicamente en determinar que documentos de una colección contienen las palabras clave insertadas por el usuario, lo cual no es suficiente para satisfacer la necesidad de información del usuario. De hecho, un usuario de un sistema de IR está interesado en obtener información sobre un tema determinado.

El objetivo de este trabajo es probar si la expansión de consultas utilizando el modelo del usuario mejora la ejecución de la recuperación. Los usuarios de sistemas de recuperación que usan la coincidencia de palabras como base de la recuperación se enfrentan con el reto de expresar sus consultas con palabras de los

vocabularios de los documentos que desean recuperar. Esta dificultad es grave en grandes bases de datos de texto debido a que dichas bases de datos contienen muchas expresiones diferentes para referirse al mismo concepto. La habilidad para recuperar documentos de estas bases de datos es crucial en un amplio rango de aplicaciones: recuperar documentación para la ayuda en casos legales, facilitar la organización y recuperación de correspondencia y formularios en una oficina, filtrar fuentes de noticias para encontrar artículos de interés, encontrar pasajes relevantes dentro de un conjunto completo de manuales de un sistema complejo para un problema particular a mano, etc. Un método de aligerar la carga del usuario cuando selecciona las palabras de una consulta es que el sistema de recuperación expanda automáticamente la consulta añadiéndole términos que estén relacionados con las palabras proporcionadas por el usuario. Los nuevos términos pueden estar estadísticamente relacionados con las palabras originales de la consulta, es decir, los términos tienden a concurrir en documentos o pueden ser elegidos con la ayuda de los recursos lingüísticos.

El uso de información de ocurrencia para expandir vectores de consulta es atractivo debido a que las relaciones se pueden generar fácilmente a partir de los documentos, obviando la necesidad de recursos lingüísticos u otras herramientas adicionales, que son costosas de desarrollar y mantener. Desafortunadamente, tales métodos han tenido poco éxito en mejorar la eficacia de la recuperación cuando se

usan sin datos de relevancia. Sin embargo, métodos que explotan relaciones estadísticas pero no expanden la consulta, como el Latent Semantic Indexing [8], sí han sido más exitosos.

El uso de información lingüística como fuente de términos relacionados ha encontrado algún éxito en pequeños experimentos. Salton y Lesk [4] encontraron que la expansión mediante sinónimos mejora el rendimiento, pero los resultados de la expansión mediante términos más generales o más restringidos seleccionados de recursos lingüísticos jerárquicos eran demasiados inconsistentes para ser útiles en general, al realizarse sobre colecciones pequeñas. Wang, Vendendorpe y Evens [12] encontraron que una variedad de relaciones léxico-semánticas mejoran el rendimiento. Sin embargo, cada una de estas conclusiones se alcanzó en experimentos sobre colecciones muy pequeñas usando recursos lingüísticos de un único dominio. En cuanto a la utilidad de la expansión de consultas mediante relaciones léxico-semánticas en colecciones grandes que abarcan varios dominios, los trabajos más representativos son los que utilizan la colección de referencia TREC que comentaremos más adelante.

Los estudios previos con estas características en el uso de información lingüística han ofrecido distintos resultados. Voorhees [3] mostró que usar WordNet para la expansión de consultas no aumenta la efectividad de la recuperación de información. En un trabajo de Mandala [7] las relaciones almacenadas en WordNet se combinan con otras medidas de similitud basadas en dependencias sintácticas e información de concurrencia y sí mejoran considerablemente la eficacia de la recuperación de información.

En el trabajo de Ellen M. Voorhees [3] las consultas se expanden usando las relaciones codificadas en WordNet, recurso lingüístico que comentaremos en la sección 2, y son evaluadas contra los tópicos del 401 al 450 producidos por la octava conferencia TREC. Para evitar los efectos distorsionadores de partir de una consulta original con selección pobre de palabras, los términos originales de la consulta que se expanden fueron generados automáticamente a partir de los tópicos, seleccionado ade-

más los conjuntos de sinónimos de WordNet que se consideraba que resultaban los conceptos importantes del tópico. Por eso, los resultados obtenidos en este trabajo representan un límite superior en el rendimiento que se puede esperar en un procedimiento de selección totalmente automático que use la estrategia de expansión. Incluso en el mejor caso, la expansión no mejoró la efectividad de las consultas que eran completas inicialmente, es decir, de las consultas que contenían términos suficientemente representativos y que describían el tópico de interés con detalle. Consultas menos completas, consistentes en una sentencia simple describiendo el tópico de interés, sí mejoraban significativamente con la expansión.

En la sección 2 se explicarán de forma clara el modelo utilizado y cuál va a ser la medida de similitud entre las consultas y los documentos para determinar cuáles son relevantes y cuáles no. Además esta medida será útil para determinar un orden (rankear) de los documentos relevantes. En la sección 3 se describirá brevemente WordNet. En la sección 4 se presentará la aplicación desarrollada para la formulación de las consultas y diseño de los experimentos. En la sección 5 se describen los experimentos y sus resultados. Las conclusiones son presentadas en la sección 6.

2. Modelo de usuario

Uno de los problemas más importantes en recuperación de información consiste en formular la consulta para que plasme adecuadamente la necesidad informativa del usuario. Aparte de los requerimientos del sistema para formalizar la consulta, el mayor problema consiste en determinar el conjunto de palabras que expresen semánticamente esa necesidad. El problema se agrava debido al efecto de inconsistencia en la asignación subjetiva de términos a conceptos. Figuras como la sinonimia o la polisemia (u otras menos importantes, como la homonimia, antonimia, la hiperonimia, la hiponimia etc) hacen que el mismo concepto pueda expresarse con palabras diferentes y una misma palabra pueda aparecer en documentos que tratan sobre temas distintos. En

esta situación no es de extrañar que el usuario tenga que replantear su consulta para obtener mejores resultados.

Se han propuesto diversos mecanismos para construir la nueva consulta. En general, en todos ellos se realiza una ampliación de nuevos términos a la consulta inicial y un recalcu de la importancia de cada término. Esto es lo que se conoce como expansión de consulta. Se pretende ampliar el número de términos que mejor definan la necesidad informativa del usuario de acuerdo a la colección documental y al modelo de recuperación utilizado.

El interés para agilizar la expansión de las consultas se centra en consultas con muy pocos términos, pues las consultas largas suelen proporcionar buenos resultados de recuperación, al incluir más términos comunes con los documentos relevantes. De hecho, la mayor parte de las consultas que se realizan en buscadores y sistemas de información en Internet tienen de uno a tres términos [2].

2.1. Descripción del modelo

Supongamos que T , D , N son campos del tópico t de TREC [11] de la consulta q , donde T es el título, D es la descripción y N es la narrativa, que se pueden representar por $T + D + N$, eliminando los campos $N + D$, quedaríamos con sólo el título, debido a que según los experimentos realizados por Voorhees [3] demostraron que las consultas expandidas obtenidas a partir de consultas originales, más cortas, sí consiguen mejoras significativas con respecto a las originales en su versión más larga. Además los experimentos de Voorhees demostraron, por lo tanto, que la expansión mediante relaciones léxico-semánticas proporcionan un beneficio pequeño cuando el usuario proporciona una consulta detallada. Aún así, los usuarios frecuentemente no proporcionan una consulta detallada. En este caso, las relaciones léxico-semánticas tienen el potencial de mejorar las consultas iniciales, aunque la consulta expandida es poco probable que sea tan efectiva como una consulta mejor formulada y proporcionada por el usuario.

Sea $\vec{q}_1 = \langle A_0, B_0, C_0 \rangle$ el vector de la consulta no expandida con sólo el título, donde A_0 ,

B_0 y C_0 son términos pertenecientes a q_1 en los cuales A_0 y C_0 son nombres. Supongamos también que a partir de la expansión eligiendo sólo los nombres que aparecen en la descripción textual del tópico utilizando el recurso lingüístico obtenemos los términos siguientes: A_1 , A_2 y A_3 como sinónimos relacionados con A_0 y C_1 , C_2 , C_3 y C_4 como sinónimos relacionados con C_0 , entonces el nuevo vector de la consulta expandida quedará del siguiente modo: $\vec{q}_2 = \langle A_0, B_0, C_0, A_1, A_2, A_3, C_1, C_2, C_3, C_4 \rangle$. Supongamos también que en la consulta: $\vec{q}_2 = \langle A_0, B_0, C_0, A_1, A_2, A_3, C_1, C_2, C_3, C_4 \rangle$ se ha seleccionado los términos A_1 , C_1 y C_2 como los correctos la consulta quedará de la siguiente manera: $\vec{q}_3 = \langle A_0, B_0, C_0, A_1, C_1, C_2 \rangle$ con esto se pretende que si en una consulta un termino original tiene diversos significados, se supone que aquellos que son distintos del correcto no influyen en eficacia de la recuperación. Supongamos además que cada documento y cada consulta son representados por un vector de la frecuencia del término $\vec{d} = \langle x_1, x_2, \dots, x_n \rangle$ y $\vec{q} = \langle y_1, y_2, \dots, y_n \rangle$ respectivamente, donde n es el número total de términos, o tamaño del vocabulario y x_i , y_i son las frecuencias del término t_i en d y q respectivamente.

Dada una colección C , la frecuencia del documento inverso (*idf*) de un término es dado por $\log(N/n_i)$, donde N es el número total de documentos en C y n_i es el número de documentos con el término i . Todos los términos en la consulta son pesados por la fórmula de pesado *TFIDF* heurística. Esto es los vectores pesados por d y q son:

$$\begin{aligned} \vec{d} &= \langle tfd(x_1)idf(t_1), \dots, tfd(x_n)idf(t_n) \rangle \\ \vec{q} &= \langle tfq(y_1)idf(t_1), \dots, tfq(y_n)idf(t_n) \rangle \end{aligned}$$

$RawTF = TF(t_i, d_j)$ indica la frecuencia del término t_i en el documento d_j e *idf* = $\log(N/n_i)$ la frecuencia del documento inverso. Si el mínimo es 0 entonces $n_i = N$, es decir, el término t_i aparece en todos los documentos; si el máximo es $\log(N)$, entonces $n_i = 1$, es decir, el término t_i aparece en un documento, siendo t_i es el número de términos índices.

La frecuencia inversa depende de la colección, necesitamos N (número de documentos

pero es un entero que depende de la colección). Finalmente el esquema de pesado *tf/idf* asigna el siguiente peso al término t_i en el documento d :

$$W_{i,d} = \text{RawTF} * \text{IDF} = \text{RawTF} * \log(N/n_i)$$

La función *TF* de la consulta está definida de forma similar como sigue:

$$W_{i,q} = \text{RawTF} * \text{IDF} = \text{RawTF} * \log(N/n_i)$$

La puntuación de un documento d con la consulta q viene dada por la fórmula

$$S(\vec{d}, \vec{q}) = \sum_{i=1}^n \text{tf}_d(x_i) \text{tf}_q(y_i) \text{idf}(t_i)$$

Una vez obtenida la consulta expandida, el campo título seleccionado es indexado utilizando las rutinas estándares del Sistema de Recuperación de Información Lemur, se computa un rank de documentos para cada consulta. Para cada consulta los documentos están ranked en orden decreciente de similitud. La puntuación de un documento d con la consulta q es dada por una expresión matemática a cima representada y posteriormente una vez obtenidos los resultados de la consulta expandida y de los distintos experimentos contemplados se procederá a la evaluación y la comparación entre ellas con las consultas sin expandir.

3. WordNet

WordNet es un diccionario MRD (Machine Readable Dictionaries), para el idioma inglés (ver [1, 9, 12]) convirtiéndose en uno de los recursos más valiosos para el procesamiento del lenguaje natural (PLN). El desarrollo de WordNet se inició en 1985 y en el Laboratorio de Ciencias Cognitivas de la Universidad de Princeton bajo la dirección del Profesor de Psicología George A. Miller. Este recurso posee una base de datos que agrupa las palabras en conjuntos de sinónimos llamados synsets y provee definiciones, comentarios y ejemplos de uso de estas palabras y sentido de las mismas. De esta manera, combina los elementos de un

diccionario (definiciones y algunos ejemplos) y los de un tesoro (sinónimos), y crea un apoyo muy importante para el análisis automático de textos y palabras.

4. Generación de consultas

El procedimiento de expansión usado en este trabajo fue el siguiente: A partir de los ejemplos de necesidades de información que la TREC incluye se realizaba una selección automática de términos. Para cada uno de esos términos, y mediante la ayuda de WordNet, se seleccionaba, también de forma automática, un conjunto de sinónimos de cada uno de estos términos originales que se consideraban del sentido más correcto para añadir al vector de la consulta. Todos los sinónimos contenidos dentro de un conjunto de sinónimos de la cadena son añadidos a la consulta. Las palabras más comunes en el idioma o stop words, como por ejemplo of, se quitan, y a las palabras restantes no se les ha aplicado un procesamiento morfológico consistente en quedarse con la raíz de la palabra (stemming). Para implementar la aplicación se han utilizado las tecnologías asociadas a la plataforma Visual Basic .Net. que nos permite probar las diferentes opciones de la expansión.

Para la realización de la interfaz de usuario se ha optado por una interfaz de ventanas bajo el sistema operativo Windows XP. La capa modelo está implementada en Basic .Net. A la hora de definir el ejercicio o experimento existen un gran número de posibilidades cuyo significado se detalla a continuación: En primer lugar, es necesario indicar el rango de tópicos sobre los que se realizará el ejercicio. Para ello, en el primero de los campos del formulario (Introducir tópicos (151-550)) se introduce un rango en la formulación min-max, donde min será el número de tópico de comienzo y max el número de tópico a usar en el ejercicio. Los números de los tópicos deben estar comprendidos entre 151 y 550 (ver figura 1).

Después se selecciona el algoritmo de stemmer. Seleccionar stemmer para consulta a aplicar sobre las palabras, para pasar a formar parte de las consultas. Se puede seleccionar no

Figura 1: Interfaz de la aplicación que nos permite definir las opciones de la expansión.

aplicar ninguno, o bien aplicar el algoritmo de Porter [5]. Si se elige no aplicar ninguno, las palabras de las consultas irán completas, si se elige el algoritmo de Porter aparecerán únicamente las raíces obtenidas por este algoritmo.

La siguiente posibilidad Seleccionar Stemmer para acceso a WordNet nos permite indicar si queremos que a la hora de buscar una palabra en WordNet se busque de manera exacta y sólo se encuentre si está incluida exactamente o si se desea aplicar el procesamiento morfológico de WordNet encontrando así también las palabras cuya raíz aparece en WordNet aunque no aparezca en sí misma. Según la opción que se escoja en Elija la taxonomía en la que buscar la búsqueda de significados devolverá los resultados de buscar las palabras sólo como nombres, sólo como verbo, sólo como adjetivo, sólo adverbio o buscando todos los significados de todas las palabras en todas las taxonomías.

La siguiente sección Palabras a usar en la expansión permite indicar que palabras se añadirán a cada consulta en la expansión pudiendo-

se expandir con sinónimos, hipónimos de primer nivel, o con ambos tipos a la vez, ya que se permite la selección múltiple. Como las consultas se generan a partir de los tópicos de los títulos la siguiente opción es Tags a usar, donde se debe indicar a partir de que partes de un tópico se deben generar, pudiéndose escoger entre el Título (Headline), la Descripción (Head) o la Narrativa (Text) o cualquier combinación de ellas. Con estos tags de los tópicos (términos a usar antes de expansión) se seleccionan las partes que forman la consulta sin expandir, para ello se puede marcar una selección manual de los términos o una selección automática indicando que se use, todos los términos de los tags.

A los términos seleccionados se les puede aplicar un filtro o no según lo escogido en la selección Stop List. En ella indicaremos si no se aplicará ningún tipo de filtrado, o bien si se eliminarán aquellas palabras consideradas como muy comunes por las herramientas de recuperación de información Smart. La última opción Tipo de expansión de los términos permite escoger el tipo de expansión pudiéndose expandir mediante términos de los synsets escogidos como correctos.

Al terminar de cumplimentar el formulario, pulsando el botón Seleccionar crearemos el ejercicio del que nos aparecerá un breve resumen con sus datos y que ya estará disponible para ser realizado. Este formulario será el mismo para todos los tópicos que haya que procesar en el ejercicio. En primer lugar, la página sitúa al usuario en los tópicos que se van a procesar, identificándolos con sus números y mostrando los campos del título, descripción y narrativa. Esto se hace así independientemente de que la selección se efectúe a partir de un solo campo del tópico. A partir de esta información el usuario podrá conocer el significado de las palabras seleccionadas dentro del contexto del tópico. Debajo de estos campos aparece una lista con estas palabras seleccionadas y sus posibles significados y sinónimos obtenidos de WordNet. Según el tipo de ejercicio, el usuario podrá seleccionar automáticamente el significado correcto para cada palabra.

5. Experimentos y resultados

Para la evaluación de la expansión se ha utilizado el sistema de recuperación de información Lemur [10] y la colección de referencia TREC. Concretamente, en todos los experimentos realizados que se comentarán a continuación, se tomaron los tópicos TREC [11] numerados desde 401 a 450 para obtener cincuenta consultas de pruebas en cada experimento. En cuanto a la colección de documentos se usó la colección Small Web de TREC-8 “WT2g” que contiene 250.000 documentos aproximadamente, un conjunto de documentos relevantes proporcionados por especialistas, y además se utilizaron las medidas de recall y precisión para determinar los resultados de la recuperación.

El objetivo de definir el experimento fue probar si la expansión de la consulta original y la selección del significado correcto producen mejoría en la recuperación de la información. Como se puede observar en la figura 2, los resultados de la expansión de consulta con la selección del significado correcto fueron mejor con una precisión media no interpolada de 0.205 que la expansión de la consulta sin la selección del significado correcto con una precisión media no interpolada de 0.164. Por otro lado la consulta expandida con la selección del significado correcto mejoró ligeramente los resultados de la consulta original con una precisión media no interpolada de 0.243.

Los experimentos realizados consistieron en seleccionar automáticamente los términos relacionados para cada una de las palabras de las consultas originales y efectuar dos expansiones, una con los sinónimos para el significado correcto de cada palabra y otra con los sinónimos para los dos tipos de significados. Los términos relacionados se añaden, en este y en todos los experimentos, directamente al término original y no como términos independientes al final de la consulta. Es decir, dada una consulta $\langle A, B, C \rangle$ el término A_1 relacionado con A y el término B_1 relacionado con B la consulta expandida, quedará del siguiente modo $\langle A, A_1, B, B_1, C \rangle$. Los términos relacionados seleccionados fueron los sinónimos para

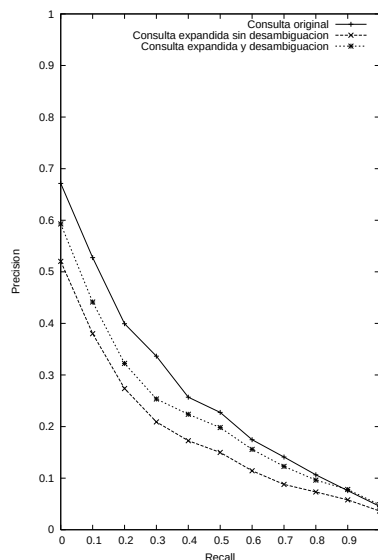


Figura 2: Gráfico de Recall-Precisión de las consultas original, expandida sin desambiguación del significado de la palabra y expandida con desambiguación del significado.

el significado correcto de cada palabra; el método desambiguación utilizado fue el basado en el diccionario, utilizando el recurso lingüístico WordNet bajo el siguiente algoritmo:

- Adquirir desde WordNet los conjuntos de sinónimos de las palabras a desambiguar.
- Determinar la coincidencia entre el contexto de las palabras a desambiguar y los conjuntos de sinónimos.
- Escoger el significado correcto de las palabras en un texto, dado el contexto de forma automática.

Este algoritmo aumentó la eficacia de 20 % con relación a la consulta expandida sin la selección del significado correcto y 16 % con relación a la consulta original.

6. Conclusiones

Con este trabajo hemos mostrado una aplicación que permite hacer la experimentación

con expansión de consultas en el ámbito de la recuperación de información mediante el uso de recurso lingüístico WordNet y bajo el modelo del usuario, utilizando una metodología de evaluación basada en la simulación. Para implementar estas funcionalidades de formulación y expansión de consultas se ha accedido a la información almacenada en WordNet. Una vez obtenidas las consultas expandidas de los distintos experimentos contemplados se ha procedido a su evaluación y la comparación entre ellas y con las consulta sin expandir. Los resultados de la evaluación son positivos con relación a la consulta expandida sin la selección del significado correcto y han permitido extraer algunas conclusiones importantes:

- Por un lado utilizando diferentes técnicas de expansión los resultados mejoraron ligeramente frente a la consulta original.
- Por otro lado los resultados demuestran que es preciso hacer uso de la desambiguación del significado correcto de la palabra para sacar el máximo partido a la expansión de la consulta con WordNet.

Referencias

- [1] A. Miller, R.Beckwith , C.Fellbaum, D. Gross y K. J. Miller, Introduction to WordNet: An on-line lexical database, en International Journal of Lexicography, Vol.3, No.4, 1990, pp.235-312.
- [2] D. Wolfram, A.Spink, B. J: Janses y T.Saracevic, Vox populi: The public searching of the web, journal of the American Society for Information Science and Technology 52(12), 1073-1074 (2001).
- [3] E. M. Voorhees (1994): Query expansion using Lexical -Semantic Relations, Proceedings of ACM-SIGIR' 94, 61-69.
- [4] G. salton, M. E. Lesk (1971): Computer evaluation of indexing and text processing, The Smart Retrieval System: Experiments in Automatic Document Processing, 143-180, Prentice-Hall, Inc. Englewood Cliffs, New Jersey.
- [5] M.F.Porter (1997): An algorithm for suffix stripping, Readings in information retrieval, Morgan Kauffman Publishers, 313-316.
- [6] Miller, R. BeckWith, C. Fellbaum, D. Gross y K. J. Miller: Introduction to WordNet: An on-line lexical database, en International Journal of Lexicography, vol. 3, No. 4, 1990, pp. 235-312.
- [7] R. Mandala, T. Tokunaga, H. Tanaka (1999): Combining multiple evidence from different types of thesaurus, Proceedings of ACM-SIGIR' 99, 191-197. (5)
- [8] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman: Indexing by latent semantic analysis, Journal of the American Society for Information Science, 41(6), 391-407.
- [9] S. M. Harabagiu, G. A. Miller y D.I. Moldovan (1999): WordNet 2 -A Morphologically and Semantical Enhanced Resource, en Proceedings of the SIGLEX Workshop.
- [10] Sitio web del LEMUR: <http://www-2.cs.cmu.edu/~lemur>
- [11] Sitio web del TREC NIST: <http://trec.nist.gov>
- [12] Yin-Chen Wang, J. Vandendorpe, M. Evens (1985): Relational thesauri in information retrieval, journal of the American Society for Information Science, 36(1): 15-27.

UML+OCL: Una Alternativa para Modelar Servicios Web Semánticos

Diana M. Sánchez, César J. Acuña, José M^a Cavero, Esperanza Marcos

Universidad Rey Juan Carlos. Dep. de Informática, Estadística y Telemática.
Escuela Superior de Ciencias Experimentales y Tecnología. Cl. Tulipán, s/n. Móstoles 28933
{diana.sanchez,cesar.acuna,josemaria.cavero,esperanza.marcos}@urjc.es

Resumen

El auge actual de la Web Semántica y en particular de los Servicios Web Semánticos (SWS), necesita de técnicas que faciliten la inclusión de esta nueva tecnología en las aplicaciones informáticas actuales. Por tanto, la implementación de los SWS requiere que las metodologías de desarrollo Web actuales se adapten para soportar el desarrollo sistemático de SWS. Con esta motivación, en trabajos anteriores hemos presentado un perfil UML para el modelado de Servicios Web Semánticos basándonos en WSMO. Este perfil extiende de SOD-M, un método para el desarrollo orientado a servicios de Sistemas de Información Web enmarcado dentro de MIDAS, un marco metodológico dirigido por modelos. El objetivo de este documento es proponer mejoras a este perfil UML, mediante la inclusión de OCL para la representación de axiomas con WSMO.

1. Introduction

Desde la aparición de la Web han surgido múltiples metodologías para abordar el desarrollo de distintos aspectos de las aplicaciones Web. Conforme la Web fue evolucionando, dichas metodologías fueron adaptadas para ajustarse a dicha evolución. Por ejemplo, con el surgimiento de los servicios Web, las metodologías Web fueron adaptadas para soportar el desarrollo de los nuevos elementos re-

queridos en las aplicaciones que interactuaban con Servicios Web, agregando técnicas y modelos específicos para la generación de documentos WSDL, para el modelado de la composición de servicios, etc. Actualmente asistimos al surgimiento de los servicios Web semánticos (SWS), que expanden las capacidades de los servicios Web tradicionales mediante el agregado de descripciones semánticas que permiten que las tareas de descubrimiento, invocación y composición (entre otras) puedan automatizarse [4]. El surgimiento de los SWS requiere el desarrollo de nuevos elementos en las aplicaciones que interactúan con ellos. Sin embargo, las metodologías actuales no proponen técnicas y modelos específicos para el desarrollo de estos nuevos elementos requeridos por las aplicaciones basadas en SWS.

Al incluir el desarrollo de los SWS en las metodologías Web actuales se obtienen dos grandes beneficios: por un lado, se permite que los SWS se puedan desarrollar sistemáticamente y, por el otro lado, se fomenta la adopción de esta nueva tecnología y de las ventajas que esta aporta. Desafortunadamente, los lenguajes de especificación semántica utilizados para describir los SWS tienen una curva de aprendizaje que resulta muy empinada para la media de los desarrolladores, lo que representa una barrera importante para la adopción de esta tecnología. Para tener éxito en el desarrollo de aplicaciones reales y para que su uso se extienda es necesario que el desarrollo de los SWS se

acople a las tendencias actuales de ingeniería del software, como es el caso de MDA [8].

Teniendo esto en cuenta, en [1] hemos presentado un perfil UML para el modelado de los nuevos elementos requeridos por las aplicaciones basadas en SWS. Dicho perfil UML es una extensión SOD-M [2] incluido, además en MIDAS, un marco metodológico dirigido por modelos para el desarrollo orientado a servicios de sistemas de información Web (SIW). La motivación para el desarrollo del perfil UML fue la carencia de técnicas específicas y enfoques metodológicos para el desarrollo de SWS. Además, si tenemos en cuenta que proponemos incluir el desarrollo de SWS en un marco metodológico dirigido por modelos, podemos lograr que el desarrollo de estos se vea beneficiado por las ventajas del enfoque MDA. Esta motivación se refuerza en el hecho de que el desarrollo de SWS es uno de los campos de investigación con más auge actualmente y donde podemos encontrar varios proyectos con empresas involucradas en la creación de soluciones reales. Esto nos lleva a pensar que los SWS tienen grandes posibilidades de ser adaptados a las aplicaciones Web en corto tiempo. Por tanto, se hace necesario contar con técnicas específicas para el desarrollo de SWS como las que pretendemos presentar en este documento.

La primera versión del perfil UML que se propone, permite modelar SWS de acuerdo a la propuesta de WSMO [6]. En WSMO, la definición de SWSs requiere la construcción de cuatro tipos de elementos nuevos, las ontologías, los servicios Web, las metas (goals) y los mediadores. Así, el perfil UML contiene cuatro modelos a nivel PSM uno para cada elemento de modelado WSMO. Partiendo de estos modelos en UML y aplicando las reglas de transformación apropiadas es posible obtener la especificación de los elementos nuevos en WSML [7, 12]. El perfil UML presentado en [1] permite modelar los cuatro elementos fundamentales de WSMO; sin embargo un aspecto -los axiomas WSMO- se modelan mediante notas de UML dentro de las cuales se introduce el código WSML que especifica los axiomas, de esta forma los desarrolladores aun necesitan conocer WSML para poder especificar SWS.

Este trabajo se centra en mejorar perfil UML para crear una forma de desarrollar Servicios Web Semánticos totalmente compatible con UML. Para ello se propone complementar este perfil usando OCL [9] para la representación de los axiomas WSMO. OCL ha surgido dentro de UML para dar completitud a los modelos creados en UML, por tanto OCL puede ser un lenguaje mucho más familiar para los desarrolladores actuales de software, de lo que podría ser WSML. Para determinar si OCL contiene toda la capacidad semántica necesaria para la representación de axiomas WSMO, analizaremos la relación entre WSML y OCL apoyados en tres casos de estudio.

Con el fin de exponer este tema, hemos estructurado este documento de la siguiente manera: en la sección 2 se definen los conceptos previos que enmarcan nuestro trabajo. La sección 3 presenta de forma detallada los tres casos de estudio en los que pretendemos evaluar la capacidad de OCL para representar axiomas. Finalmente, en la sección 4 presentamos las conclusiones de este trabajo así como los trabajos futuros en esta línea de investigación.

2. Conceptos Previos

En esta sección presentamos algunos conceptos que sirven para contextualizar nuestro trabajo. En primer lugar, la sección 2.1 presentará una breve descripción de WSMO y en la sección 2.2 introduciremos el Perfil UML que hemos desarrollado y que buscamos mejorar con el uso de OCL.

2.1. WSMO: Web Services Modeling Ontology

WSMO es una propuesta de DERI [3] para la descripción de SWSs. Define las especificaciones ontológicas para los cuatro elementos centrales de SWS, a saber:

Ontologías: proporcionan la terminología utilizada por otros elementos de WSMO para describir los aspectos pertinentes de los dominios de discurso.

Servicios Web: describen las entidades computacionales que proporcionan el

acceso a servicios que proporciona algún valor en un dominio. Estas descripciones comprenden las capacidades, interfaces y el trabajo interno del servicio Web.

Metas: representan los deseos de los usuarios, que podrían ser satisfechos mediante un servicio Web.

Mediadores: son componentes de WSMO para resolver los problemas de la interoperabilidad entre elementos diferentes.

En la propuesta de WSMO también se integra WSML un lenguaje de descripción semántica cuyo objetivo es definir formalmente los elementos definidos en WSMO y WSMX que provee un entorno para el descubrimiento, selección, mediación, invocación e interoperabilidad de SWS. Una descripción detallada de WSML se aborda en la sección 3.

2.2. Un perfil UML para el modelado de Servicios Web Semánticos

El perfil UML presentado en [1] y que resumimos en esta sección pretende modelar SWS siguiendo la propuesta WSMO. Para ello cuenta con cuatro modelos, uno por cada elemento de WSMO, es decir, un modelo de ontología, un modelo de servicios Web, un modelo de metas y un modelo de mediadores. La figura 1 muestra la relación que existe entre estos modelos enmarcándolos dentro de la estructura de 4-niveles planteada por la arquitectura MDA. El metamodelo que define WSMO ha sido especificado usando MOF [10]. El perfil presentado refina este metamodelo mediante cuatro nuevos metamodelos los cuales se encargan de la construcción de metas, ontologías, servicios web semánticos y mediadores; por tanto, estos cuatro nuevos metamodelos quedarán al mismo nivel del metamodelo WSMO (nivel M2). Cada metamodelo permitirá crear modelos (de nivel M1) los cuales representarán a su vez los elementos de WSMO.

Para hacer uso de las capacidades gráficas de UML cada uno de estos modelos se corresponde con un perfil UML que define los nuevos estereotipos, valores etiquetados y restricciones. Estos perfiles se encuentran ubicados en

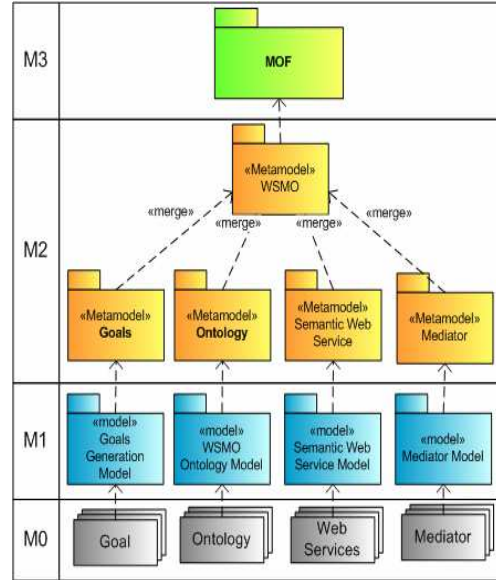


Figura 1: Arquitectura de modelos para el modelado de elementos WSMO.

el nivel M2 de la figura 1 pero, por razones de claridad, hemos omitido su representación en dicha figura. Las instancias de los modelos definidos (en nivel M1), es decir las ontologías, metas, servicios Web y mediadores concretos se ubican en el nivel M0. De aquí en adelante nos centraremos en el modelo de Ontología para explicar el modelado de elementos WSMO, además de ejemplo este modelo nos permitirá ver cómo OCL puede mejorar el perfil UML.

Para modelar una ontología WSMO hemos decidido dividir su modelado en dos modelos relacionados: el Modelo de Contexto de la Ontología y el Modelo de Contenido de la Ontología. Esta división se debe a que en WSMO la información contextual de la ontología (namespaces, ontologías importadas, etc) está claramente diferenciada de la información que representa el contenido de la ontología en sí como son los conceptos, los atributos, etc.

El modelo de contexto de la ontología extiende el diagrama de paquetes de UML y recoge información sobre el contexto en el que la ontología tiene validez. Dentro de la informa-

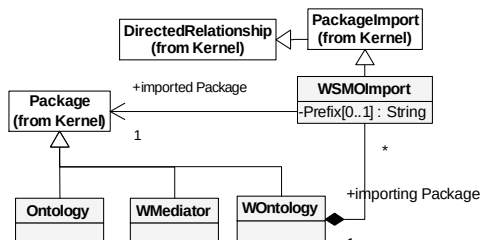


Figura 2: Metamodelo de Contexto de Ontología.

ción que podemos representar en este modelo podemos observar la inclusión de namespaces, ontologías o mediadores, que ya han sido creados y que son usados dentro de la ontología. La figura 2 muestra el Metamodelo de Contexto de Ontología.

Por su parte, el modelo de contenido de ontología representa los elementos que son modelados y especificados en la ontología: clases, propiedades, instancias, axiomas, etc. Este modelo extiende el Diagrama de Clases de UML. La figura 3 muestra el metamodelo correspondiente a un modelo de contenido de una ontología.

En el metamodelo de contenido de la ontología, los axiomas eran originalmente representados mediante una clase UML estereotipada con <<Axiom>> y con la definición del axioma como un valor etiquetado de tipo string. El valor etiquetado contiene el axioma escrito en código WSML (véase el área resaltada por las líneas punteadas en la figura 3). Para escribir el axioma, el desarrollador debe conocer WSML. Este, es precisamente el aspecto que se pretende resolver con la utilización de OCL y que se discute en la sección siguiente.

3. OCL para la representación de axiomas WSMO

En esta sección discutiremos si es posible completar el perfil UML presentado en [1] utilizando OCL de modo tal que permita la representación de axiomas. La motivación para este trabajo es permitir, a los desarrolladores familiarizados con UML, desarrollar SWS sin necesidad de tener que aprender lenguajes espe-

cíficos de descripción semántica como WSML.

Los axiomas, en el contexto de WSMO, se expresan en WSML y son expresiones lógicas que ayudan a especificar de manera formal uno o varios elementos de un SWS. Por tanto, uno de los requisitos principales a la hora de encontrar un mecanismo de representación de axiomas es buscar una herramienta que nos permita representar todo lo que actualmente se puede representar con WSML.

Algunas de las utilidades dadas a los axiomas dentro de la propuesta WSMO son:

- Restringir los valores de un atributo.
- Definir o restringir la forma en la cual un concepto puede ser extendido.
- Extender relaciones o definir funciones.
- Definir las partes de la habilidad de un servicio Web (preconditions, posconditions, assumptions y effects).

Por otro lado, OCL ha sido desarrollado para dar completitud y mayor expresividad a un modelo UML. Una expresión escrita en OCL se usa para especificar una restricción o condición que actúa sobre algún elemento del modelo y que deberá cumplirse a la hora de desarrollar el sistema [9]. Algunos de los usos de OCL son:

- Especificar condiciones que serán invariantes en clases, tipos o estereotipos.
- Describir precondiciones o poscondiciones dentro de operaciones o métodos de un elemento de un modelo.
- Definir restricciones sobre operaciones.

OCL es un producto de la familia UML, concebido con la idea de ser fácil de escribir y leer para alguien familiarizado con este entorno. Por tanto, una de las grandes ventajas de OCL es que el usuario no necesitará tener fuertes conocimientos matemáticos o lógicos para poder escribir restricciones.

WSML ha sido desarrollado teniendo como punto de partida tres formalismos: la Lógica Descriptiva, la Lógica de Primer Orden y la

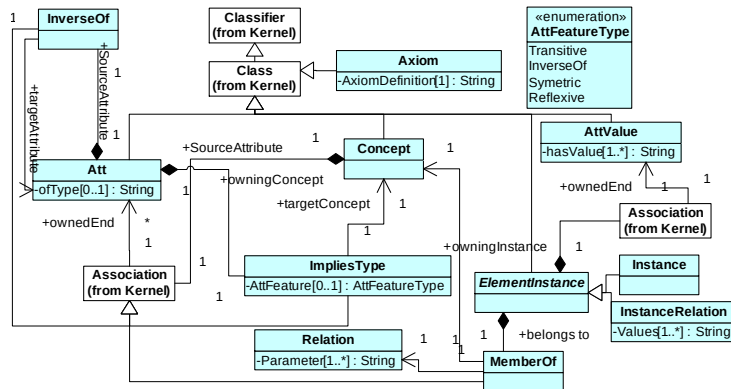


Figura 3: Metamodelo de Contenido de Ontología.

Programación Lógica. Este lenguaje presenta cuatro variantes y cada variante está basada o extiende una parte de estos formalismos. Por ejemplo, WSMML-Core (la versión básica de WSMML) está basado en la Lógica Descriptiva, que a su vez es una parte de la Lógica de Primer Orden que describe tipos. Así, el resto de las versiones de WSMML lo que buscan es extender el lenguaje hacia la Lógica de Primer Orden o hacia la Programación Lógica de acuerdo al nivel de expresividad que se desee [12]. Por su parte, OCL es un lenguaje basado en Lógica de Primer Orden y Teoría de Conjuntos. Estas características permiten ver que tanto WSMML como OCL están unidos por la Lógica de Primer Orden de la cual heredan un bloque de operaciones básicas común, una notación similar y unos recursos para la definición de los principales elementos de cada lenguaje: los conceptos en el caso de WSMML y los conjuntos en el caso de OCL. Además, si tenemos en cuenta las listas de los usos de los axiomas en WSMO con la de los usos de OCL para un diagrama UML vemos que existe una gran similitud, por lo que podemos considerar a OCL una opción para expresar axiomas.

Como hemos dicho, la principal característica que debemos tener en cuenta a la hora de elegir el mecanismo para expresar axiomas es que se puedan expresar todos los aspectos que son posibles expresar en WSMML. Como método para evaluar la viabilidad de escribir

axiomas en OCL, proponemos el análisis de un conjunto de casos de estudio. En este documento describiremos tres de estos casos. Los casos elegidos se han considerado representativos de los diferentes usos de los axiomas en WSMO. En todos los casos partiremos de una porción de código WSMML que será convertido a OCL buscando no perder expresividad. El estudio de estos casos nos permitirá establecer, al menos de forma empírica, si es posible utilizar OCL para la representación de axiomas tal como lo hace WSMML (siguiendo WSMO).

3.1. Casos de Estudio

A continuación presentamos tres casos de estudio. En cada caso el axioma utilizado cumple un objetivo diferente dentro del Modelo de Ontología presentado previamente. Como punto de partida de cada caso de estudio se presenta un axioma declarado en código WSMML. A partir de este código WSMML, aplicamos el perfil UML que hemos presentado en la sección 2.2 y que nos ayudará a representar los conceptos y atributos que se encuentran involucrados en el axioma. Finalmente y basados en la representación hecha con el perfil, presentamos el código OCL que representará el axioma deseado sin perder su significado. Estos casos de estudio se han tomado del documento WSMML Primer [13], un documento escrito por los creadores de WSMML en el que se sintetiza la funcio-

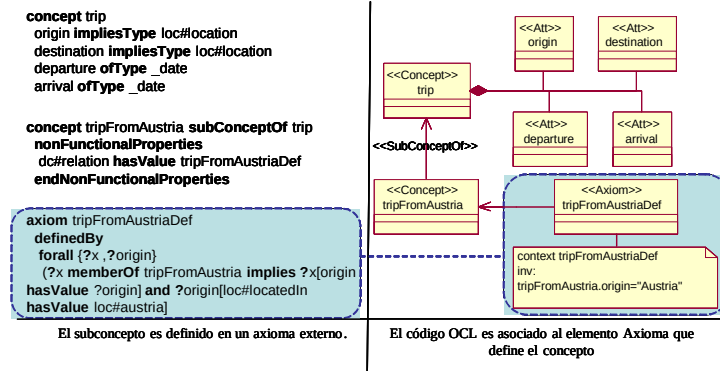


Figura 4: Un axioma usado para especificar conceptos.

nalidad de WSMO y se dan algunos ejemplos prácticos de esta propuesta.

Caso 1: Axiomas usados para definir o restringir un concepto. Este caso muestra cómo se usan los axiomas para hacer más explícito el significado de un concepto. El ejemplo representa un concepto denominado `tripFromAustria` que es declarado como un subconcepto del concepto `trip`. El concepto `trip` está compuesto por cuatro atributos: los atributos `origin` y `destination` de tipo `location`, y los atributos `arrival` and `departure` de tipo `date`. El tipo `location` ha sido definido en una ontología externa que ha sido importada a esta ontología y a la cual se hace referencia usando la expresión `loc`. En este ejemplo, la forma de especificar la característica distintiva del nuevo concepto `tripFromAustria` será mediante un elemento Axioma. La figura 4 representa el código WSMO original y el código OCL equivalente que se propone.

Caso 2: Axiomas usados para especificar una relación o una función. En este caso, la expresión lógica descrita dentro del axioma representa una relación. En WSMO, una relación describe la correspondencia entre un conjunto de parámetros mientras que una función es una clase especial de relación donde uno de los parámetros dados en la relación devuelve un resultado que se convertirá en el resultado

de la función [13].

El ejemplo presenta la relación `validCreditCard`. Este axioma especifica que una tarjeta de crédito se considera válida hasta que su fecha de vencimiento expire. El código WSMO y el código OCL equivalente en nuestro perfil se presentan en la figura 5. Hemos creado un elemento `<<relation>>` asociado al concepto implicado. Esta relación se especifica en el código OCL del `<<Axiom>>` con el que está asociada. En WSMO las relaciones se definen mediante expresiones lógicas en WSMO, por tanto las relaciones y los axiomas son expresados de la misma manera. De hecho, las relaciones son definidas como tipos especiales de axiomas que describen la correlación entre un conjunto de parámetros. WSMO permite codificar una relación de dos formas: a) escribiendo el código dentro de la definición de la relación o b) escribiendo el código en un axioma externo que se asocia a la relación dentro de su definición. Este ejemplo presenta una relación codificada siguiendo el tipo b, es decir, mediante un axioma externo asociado a la relación.

Caso 3: Axiomas para la definición de habilidades de SWS. Las habilidades definen la funcionalidad de un SWS en términos de precondiciones, poscondiciones, supuestos y efectos [13]. Según lo expuesto en [1], un SWS puede ser representado por los modelos

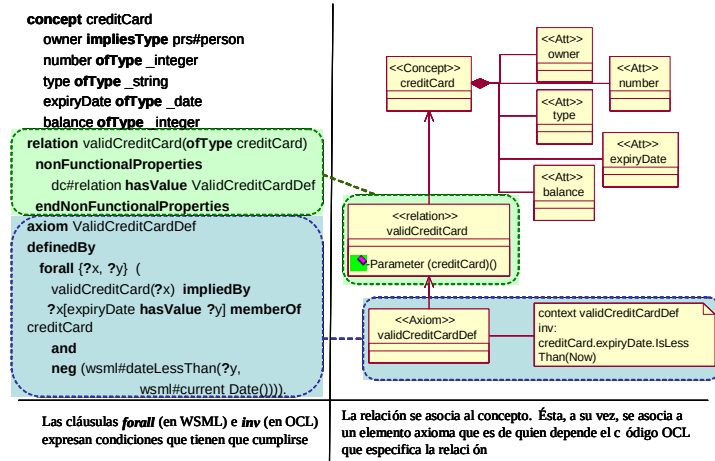


Figura 5: Un axioma usado para especificar una relación.

de contenido y contexto. El modelo de contexto recolecta información sobre el contexto de SWS, tal como la(s) ontología(s) que utiliza, donde estará(n) definidos los conceptos que usa el SWS. El modelo de contenido representa los elementos propios del SWS tales como las habilidades o las interfaces. El ejemplo que se presenta en este artículo muestra con detalle la habilidad de un SWS para reservar un billete de tren.

En WSML una habilidad esta formada por una primera parte con la definición de variables comunes y una segunda parte que especifica las precondiciones de la habilidad, otra para los supuestos o elementos (propiedades o valores) que se asumen como ciertos y por último las poscondiciones y los efectos, es decir, la forma en la que quedan las cosas después de que la habilidad se ejecuta. Para facilitar el análisis del código de la habilidad, estudiaremos cada parte por separado.

La primera parte de la habilidad estudiada nos muestra la definición de cinco variables y la especificación de las precondiciones. Algunas de las variables definidas en la zona `sharedVariables` se usan en la parte de precondiciones para que tomen los valores de los atributos del concepto `reservationRequest` y así validar la integridad de la información. Dentro del

código de la habilidad `tr\#` y `po\#` son referencias a ontologías importadas. El código WSML de esta primera parte se presenta a continuación:

```

capability BookTicketCapability
sharedVariables {
  ?creditCard, ?initialBalance,
  ?trip, ?reservationHolder,
  ?ticket}
precondition definedBy
?reservationRequest [
  reservationItem hasValue ?trip,
  reservationHolder hasValue
  ?reservationHolder
] memberOf tr#reservationRequest and
?trip memberOf tr#tripFromAustria and
?creditCard [
  balance hasValue ?initialBalance
] memberOf po#creditCard.

```

Este código WSML especifica, como precondición, que tres de los atributos del concepto `reservationRequest` y un atributo del concepto `creditCard` tengan valores válidos. En OCL para asignar un valor a una variable se utiliza la expresión `init`, y la expresión para iniciar la especificación de un conjunto de precondiciones es `pre`. A continuación presentamos cómo quedaría esta primera parte (variables compartidas y precondiciones) escrita en código OCL.

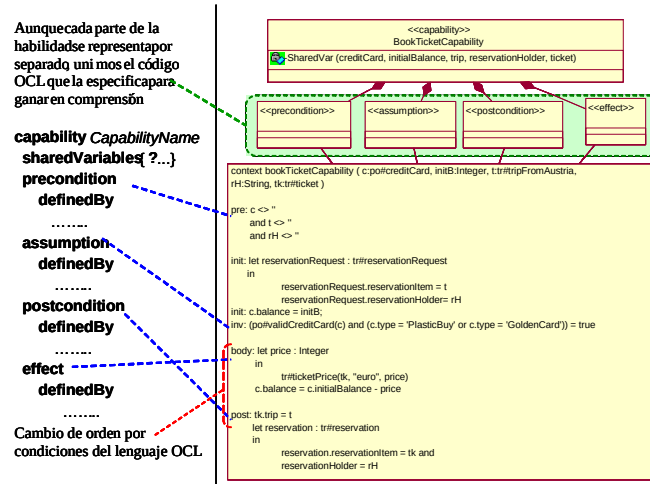


Figura 6: Axiomas usados para especificar la habilidad de un SWS.

```

context bookTicketCapability
(c:po#creditCard, initB:Integer,
t:tr#tripFromAustria, rH:String,
tk:tr#ticket)
    
```

```

pre: c <> "" and
t <> "" and
rH <> ""
    
```

```

init:
let reservationRequest: r#reservationRequest in
reservationRequest.reservationItem = t and
reservationRequest.reservationHolder = rH
    
```

```

init: c.balance = initB;
    
```

Las assumptions especifican las condiciones que se asumen como verdaderas y que deben ser reconocidas antes de iniciar la ejecución de la habilidad. En este ejemplo, se asume que la tarjeta de crédito dada deberá ser de tipo PlasticBuy or GoldenCard.

```

assumption definedBy
po#validCreditCard(?creditCard) and
(?creditCard[type hasValue "PlasticBuy"] or
?creditCard[type hasValue "GoldenCard"]).
    
```

Para expresar una condición que debe ser verdadera, OCL proporciona la palabra reservada *inv* (invariant), que ya ha sido utilizada

en los casos de estudio uno y dos. Por tanto, los supuestos para esta habilidad puede ser reescritos en OCL de la siguiente forma:

```

inv: (po#validCreditCard(c) and
(c.type = 'PlasticBuy' or
c.type = 'GoldenCard')) = true
    
```

La parte de poscondiciones muestra el resultado de la ejecución de un SWS, es decir muestra cuál será el estado del mundo después de la ejecución de la habilidad. En este ejemplo, las poscondiciones especifican que tras la ejecución de la habilidad, podremos observar un objeto tipo reserva con todos los datos de la reserva, y una reserva hecha. Como efecto de esta poscondición, se especifica que el precio del billete será deducido de la tarjeta de crédito utilizada para la transacción. El código WSML de ambas partes se presenta a continuación:

```

poscondition definedBy
?reservation memberOf
tr#reservation [
reservationItem hasValue ?ticket,
reservationHolder hasValue ?reservationHolder
] and
?ticket[trip hasValue ?trip]
memberOf tr#ticket.
    
```

```

effect definedBy
  ticketPrice(?ticket, "euro", ?ticketPrice) and
  ?finalBalance =
    (?initialBalance - ?ticketPrice) and
  ?creditCard[
    po#balance hasValue ?finalBalance
  ].

```

En OCL la cláusula `pos` sirve para especificar un bloque de poscondiciones; sin embargo, y a diferencia de WSML, en OCL este bloque deberá ir siempre al final del código. En el bloque de efectos de WSML se describen las acciones que cambiarán el estado de algunos de los elementos involucrados en el desarrollo de la habilidad. Dentro de OCL este tipo de acciones se agrupan dentro de la cláusula `body` que muestra la ejecución propiamente dicha de la función (habilidad del SWS). Si escribimos este código WSML en OCL obtenemos:

```

body: let price: Integer in
  tr#ticketPrice(tk, "euro", price) and
  c.balance = c.initialBalance - price

pos: tk.trip = t and
  let reservation: tr#reservation in
    reservation.reservationItem = tk and
    reservationHolder = rH

```

Una vez concluido el análisis de todas las partes que forman la habilidad, procedemos a unirlos para ver en conjunto cómo ésta se puede representar usando nuestro perfil. La habilidad se muestra en la figura 6. La parte izquierda de la figura muestra el esquema general del código WSML para especificar la habilidad; por razones de espacio no hemos incluido el código completo de cada sección y lo hemos reemplazado con puntos suspensivos. La parte derecha de la figura 6 muestra cómo representaríamos esta habilidad usando nuestro perfil. A diferencia de antes, el código OCL no ha sido asociado a un elemento tipo `<<Axiom>>` sino a los elementos `<<precondition>>`, `<<assumption>>`, `<<poscondition>>` y `<<effect>>` que son los que forman una habilidad en nuestro perfil.

Teniendo en cuenta que nuestros casos de estudio representan los usos más comunes de los axiomas podemos determinar que existe un

amplio conjunto de casos donde se pueda sustituir WSML por OCL para definir axiomas dentro del perfil UML para la representación de un SWS expuesto en la sección 2.

4. Conclusiones y trabajos futuros

La tecnología SWS requiere el desarrollo de nuevos componentes en las aplicaciones que interactúan con SWS. Esto, a su vez, requiere que las metodologías actuales de desarrollo Web se adapten para soportar el desarrollo de estos nuevos componentes. En trabajos anteriores hemos presentado un perfil UML que extiende SOD-M, un método para el desarrollo orientado a servicios de SIW. Este perfil UML nos permite modelar los elementos que pueden ser requeridos en la especificación de un SWS siguiendo la propuesta de descripción de SWS definida por WSMO.

Este documento muestra el trabajo que se adelanta actualmente para refinar dicho perfil agregándole la capacidad de representar axiomas (o restricciones) usando un lenguaje mucho más cercano a los desarrolladores como puede ser OCL. De este modo, usando el perfil que aquí se define es posible modelar SWS utilizando únicamente UML. Así, los desarrolladores que sólo conozcan UML podrán representar y desarrollar SWS sin necesidad de tener que aprender lenguajes de descripción semántica como WSML.

Para evaluar si OCL es adecuado para la representación de axiomas en WSMO hemos presentado tres casos de estudio. Cada uno de ellos representa una forma de uso diferente de los axiomas en WSMO. En los tres casos la transformación del código WSML a OCL ha sido posible sin perder significado. Los buenos resultados de estos casos de estudio nos llevan a determinar que existe un amplio conjunto de casos en los que será posible sustituir WSML por OCL, que sí es posible establecer una correspondencia entre estos dos lenguajes y que el siguiente paso debería ser formalizar dicha correspondencia, involucrando un estudio formal de los mismos.

Como trabajos futuros planteamos seguir refinando esta propuesta y el perfil UML, es-

tableciendo una correspondencia entre WSML y OCL que esté apoyada en una comparación formal y que nos permita crear un mecanismo de transformación entre los dos. Además, estamos trabajando en incorporar este perfil UML en M2DAT [11], una herramienta MDA que integra todas las técnicas propuestas en MIDAS para la generación semiautomática de WIS y de esta forma poder generar automáticamente código WSML a partir de modelos UML de los SWS.

Agradecimientos

Financiado parcialmente por el Ministerio de Ciencia y Tecnología de España bajo el proyecto GOLD TIN2005-0010.

Referencias

- [1] Acuña C. and Marcos E. Modeling Semantic Web Services: A Case Study. Proceedings of the 6th International Conference on Web Engineering. ACM Press. Editores D. Wolber, Neil Calder, Chris Brooks and Athula Ginije. pp 32-39, USA, 2006
- [2] De Castro V., Marcos E. and López M. A model driven method for service composition modelling: a case study. International Journal of Web Engineering and Technology 2006 - Vol. 2, No.4 pp. 335 - 353.
- [3] Digital Enterprise Research Institute. <http://www.deri.org>.
- [4] Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications. 1(2). 2002
- [5] Gannon G. and Timm J. An MDA-based Approach for Facilitating Adoption of Semantic Web Service. In Proceedings of the IEEE EDOC Workshop on Model-Driven Semantic Web (MDSW 04), September 2004.
- [6] Lausen H, Polleres A. and Roman D. (Eds). Web Service Modeling Ontology Submission. Retrieved From: <http://www.w3.org/Submission/WSMO/>, 2005
- [7] Lausen H, de Bruijn J, Polleres A, Fensel D. WSML - a Language Framework for Semantic Web Services. Position Paper for the W3C rules workshop, Washington DC, USA, April 2005. Retrieved from: <http://www.w3.org/2004/12/rules-ws/paper/44/>
- [8] Miller, J., Mukerji, J. (eds.). MDA (2001) 'OMG Model Driven Architecture', Document number ormsc/2001-07-01. Retrieved from: <http://www.omg.com/mda>.
- [9] OCL 2.0 Specification. Object Management Group (OMG). Retrieved from <http://www.omg.org/docs/ptc/05-06-06.pdf>
- [10] The Object Management Group: Meta-Object Facility, version 1.4. Retrieved from: <http://omg.org/technology/documents/formal/mof.htm>, 2002
- [11] Vara, J.M., De Castro, V. and Marcos, E. WSDL automatic generation from UML models in a MDA framework In International Journal of Web Services Practices. Volume 1 - Issue 1-2. November 2005, pp.1 - 12. 2005.
- [12] WSML Final Draft 5. DERI. Retrieved from: <http://www.wsmo.org/TR/d16/d16.1/v0.21/#sec:wsml-xml>
- [13] WSMO Primer. Retrieved from: <http://www.wsmo.org/TR/d3/d3.1/v0.1/>, 2005

Integrando Aspectos en MWACSL

Antonia M^a Reina, Miguel Toro, Jesús Torres

Universidad de Sevilla. Dep. de Lenguajes y Sistemas Informáticos.
ETSI Informática. Avda. Reina Mercedes, s/n. Sevilla 41012.
{reinaqu, mtoro, jtorres}@lsi.us.es

Resumen

MWACSL es una propuesta dirigida por modelos y orientada a aspectos para el desarrollo de aplicaciones, en la que se utilizan lenguajes específicos de dominio para expresar los aspectos a nivel independiente de plataforma. Uno de los problemas que aparecen en este ámbito es la integración de los distintos aspectos que componen el sistema. Este artículo presenta una primera aproximación a la resolución de esta problemática.

1. Introducción

En los últimos años ha resurgido con fuerza el paradigma de Desarrollo de Software Dirigido por Modelos, sobre todo gracias a propuestas como MDA (*Model Driven Architecture*) [13], o las Factorías Software (FS) [8]. Esto ha provocado que la atención se centre en los modelos, metamodelos y transformaciones, a la hora de desarrollar el software. Las transformaciones son la forma de obtener unos modelos a partir de otros.

A la vez que se ha ido expandiendo la popularidad del paradigma han ido surgiendo un conjunto de herramientas, que cada vez se encuentran en un estado de madurez mayor, para dar soporte a este tipo de desarrollo. Por otra parte, el principio de separación de competencias, que ha dado lugar a lo que en la actualidad se conoce como desarrollo de software orientado a aspectos (DSOA) [16, 17], ha aparecido como una nueva forma de descom-

poner los sistemas mejorando la modularización de los mismos para que sean más fáciles de desarrollar, adaptar y mantener [4].

Si nos fijamos en este paradigma y en MDA, una de las propuestas más populares para enfrentarse al desarrollo de software dirigido por modelos, se puede observar que las dos aproximaciones ofrecen distintas formas de separar competencias. Mientras que el Desarrollo de Software Orientado a Aspectos nos proporciona una separación horizontal (al mismo nivel de abstracción) de los distintos conceptos que conforman un sistema, MDA ofrece una separación vertical, desacoplando los conceptos que dependen de una plataforma concreta de aquellos independientes de la plataforma (Figura 1). Utilizando MDA, la separación de los conceptos tecnológicos se consigue gracias a la definición de distintos niveles de modelado: Modelos Independientes de Computación (CIM), Modelos Independientes de Plataforma (PIM) y Modelos Específicos de Plataforma (PSM).¹

En [15] se hacía un estudio de los lenguajes de modelado de aspectos y, como conclusión, se tenía que la mayoría de ellos son lenguajes de modelado de aspectos de propósito general, es decir, que con ellos se pretende modelar cualquier tipo de aspecto. Pero, además, también se obtenía que estos lenguajes son de bajo nivel, en el sentido de que definen constructo-

¹Nótese que las siglas utilizadas corresponden a la nomenclatura inglesa de estos términos, pero se ha decidido utilizar estas siglas porque están mucho más extendidas en la comunidad investigadora.

res muy cercanos a los existentes en los lenguajes de programación orientados a aspectos. Así, se apuntaba la necesidad de elevar el nivel de abstracción y la definición de lenguajes específicos de dominio para los aspectos definidos a más alto nivel, es decir, a nivel independiente de la plataforma. Así, se proponía la definición de un lenguaje específico para cada uno de los conceptos que componen la aplicación a desarrollar.

A la hora de definir estos lenguajes específicos de dominio, se pueden utilizar dos alternativas, bien extender UML mediante perfiles, bien utilizar metamodelos MOF. En [15] se apuntaba como ventaja de la utilización de perfiles era la utilización de herramientas genéricas basadas en UML. A día de hoy, y gracias a la proliferación de herramientas de modelado y metamodelado, tales como EMF [2, 5], esto ya no representa un gran inconveniente a la hora de trabajar con metamodelos, teniendo como ventaja su mayor capacidad expresiva.

El resto del artículo se distribuye como sigue. En primer lugar, en la sección 2, se da una visión general de MWACSL. A continuación, en la sección 3 se hace una exploración de las distintas opciones a la hora de integrar los aspectos. Luego, se presentan algunos trabajos relacionados, para finalizar con la conclusión del artículo y un apunte de las líneas de trabajo futuras.

2. MWACSL

Los primeros trabajos que aparecieron publicados en el área de la programación orientada a aspectos proponían una serie de lenguajes específicos de dominio para tratar con los diferentes aspectos de un sistema. Así surgieron lenguajes como COOL o RIDL [12], para tratar con los aspectos de sincronización y distribución, respectivamente. Pero los problemas que suponía el trabajar con este tipo de lenguajes hicieron que la comunidad se decantara por lenguajes de aspectos de propósito general, como AspectJ [11, 18]. Mientras que con los lenguajes específicos de dominio hace falta definir un lenguaje nuevo para cada uno de los aspectos del sistema, con los lenguajes de

propósito general se utiliza el mismo lenguaje para definir cualquier aspecto.

Gracias al auge del desarrollo de software dirigido por modelos (DSDM), que ha hecho que, por una parte, el interés se centre en los modelos a la hora de desarrollar los sistemas, y, por otra, ha puesto de relieve la importancia de los metamodelos y las transformaciones en el proceso de desarrollo. Así, están surgiendo herramientas cada vez más potentes capaces de dar soporte a la producción de software utilizando este tipo de artefactos. De esta forma, los lenguajes específicos de dominio han entrado de nuevo en la arena de la comunidad investigadora. Y es que debido a las técnicas introducidas por el DSDM, muchos de los problemas que tenían los lenguajes específicos de dominio se pueden solventar más fácilmente, cambiando el trabajo en el espacio tecnológico de las gramáticas (lo que en la bibliografía se conoce como *grammarware*) por el espacio tecnológico de los modelos (o *modelware*).

Dentro de este contexto, MWACSL es una propuesta de desarrollo de software para el ámbito de aplicaciones web, dirigida por modelos y orientada a aspectos, que propone la definición de lenguajes específicos de dominio para cada uno de los aspectos que componen un sistema. Así, podemos decir que una aplicación estará compuesta por un conjunto de modelos de aspectos, cada uno expresado en un lenguaje específico para el dominio con el que trabaja el aspecto, y un modelo primario. La parte superior de la Figura 1 muestra un esquema de este tipo de separación.

Al separar los distintos aspectos del sistema, es más fácil razonar sobre cada uno de ellos, por lo que la complejidad de la aplicación disminuye. Además, al utilizar lenguajes específicos de dominio, los conceptos con los que se va a trabajar van a estar más cercanos al dominio del aspecto, de tal forma que cuando el desarrollador trabaja con el aspecto de navegación solamente se tiene que preocupar de nodos, enlaces y las cuestiones propias de la navegación, y cuando trabaja con el aspecto de sincronización, solamente debe pensar en exclusión mutua, variables de condición, ...

Sería deseable mantener estos aspectos sepa-

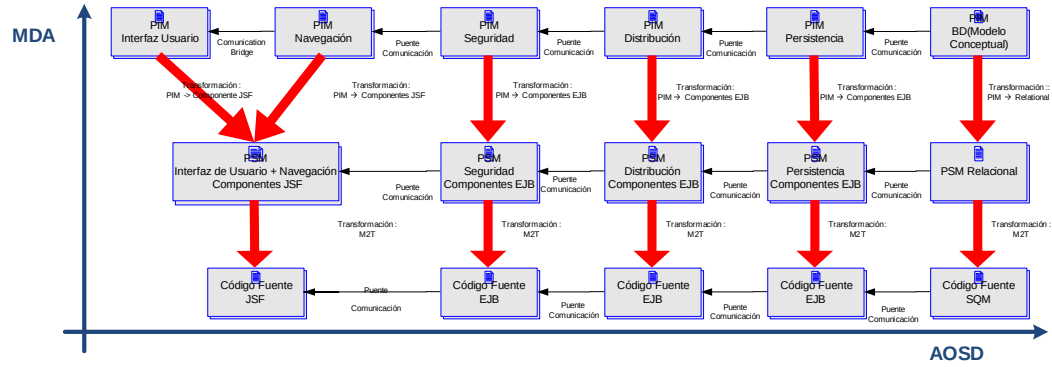


Figura 1: Esquema general MWACSL.

rados desde el diseño hasta la implementación de la aplicación, aunque esto no siempre es posible. Fijémonos, por ejemplo, en la Figura 1. En ella se necesitan dos lenguajes específicos de dominio para los aspectos de navegación y de interfaz de usuario. Sin embargo, la implementación se va a realizar utilizando el framework Java Server Faces (JSF)[1] (muchas veces este tipo de decisión la impone el cliente que ha encargado el producto software). El problema es que JSF no tiene una separación clara entre interfaz de usuario y navegación, por lo tanto, a nivel específico de plataforma, tenemos que trabajar con estos dos aspectos mezclados, tal y como se muestra en esta figura.

Para trabajar, por tanto, el diseñador dispondrá de una librería reusable de metamodelos correspondientes a los diferentes aspectos que conforman la aplicación. Con el estado actual en el que se encuentra la tecnología, no es difícil integrar estos metamodelos. En el siguiente apartado se da una visión de las alternativas disponibles a la hora de integrar estos metamodelos.

3. Integración de aspectos

Para integrar los aspectos expresados con lenguajes específicos de dominio hay dos estrategias claramente diferenciadas. La opción más evidente es la mezcla de aspectos a través de transformaciones de modelos. La operación conocida como entretejido (o *weaving*) en el

ámbito de la programación orientada a aspectos no es más que una transformación horizontal. Es decir, una transformación que toma como entrada dos o más modelos (el primario y los modelos de los aspectos, expresados todos ellos en el mismo nivel de abstracción) y devuelve un nuevo modelo, de nuevo expresado en el mismo nivel de abstracción, formado por la composición del modelo primario y los aspectos.

En el caso representado en la Figura 1, para obtener los modelos específicos de plataforma a partir de los modelos independientes de plataforma harían falta en algunos casos transformaciones oblicuas. Las transformaciones oblicuas no solamente se deben encargar de mezclar los aspectos, sino también cambian el nivel de abstracción (de PIM a PSM). Este sería el caso de los aspectos de navegación y de interfaz de usuario representados en esta Figura. La plataforma elegida para implementar la interfaz y la navegación ha sido Java Server Faces (JSF). El problema es que en JSF no hay una separación clara entre interfaz de usuario y navegación. Por lo tanto, a nivel de PSM tendremos que definir modelos conformes al metamodelo de JSF, en los que estarán mezclados interfaz y navegación. En este caso, las transformaciones de modelo a modelo para obtener los modelos conformes al metamodelo de JSF a partir de los modelos de los aspectos de navegación e interfaz serían transformaciones oblicuas.

Otra opción, en lugar de realizar este tipo de transformaciones, sería realizar primero una composición de los metamodelos que definen los distintos lenguajes específicos de dominio (en el caso de nuestro ejemplo, la interfaz de usuario y la navegación), y luego una serie de transformaciones verticales para obtener el modelo con los componentes JSF correspondientes a la interfaz y a la navegación.

En la siguiente subsección se muestran las diferentes estrategias existentes a la hora de componer metamodelos para implementar esta segunda aproximación.

3.1. Mecanismos de composición de metamodelos

En esta sección se describen las técnicas existentes para la composición de metamodelos:

1. Mediante fusión de metamodelos (*Metamodel merge*)

Cuando se componen dos lenguajes de modelado pueden aparecer colisiones de nombres entre ambos. Estos conceptos que aparecen en los dos lenguajes de modelado y se refieren a constructores que capturan a un conjunto de entidades del mundo real compartidas son los que se utilizan como puntos de enlace entre ambos lenguajes de modelado, de tal forma que los dos metamodelos se pueden fusionar en un lenguaje unificado.

2. Mediante la creación de interfaces entre metamodelos (*Metamodel Interfacing*)

Este tipo de composición se utiliza cuando los lenguajes de modelado específicos de dominio son conceptualmente distintos, pero tienen dominios relacionados. Para llevar a cabo este tipo de composición hace falta definir claramente una interfaz entre los dos lenguajes de modelado. Esta interfaz consistirá en nuevas entidades de modelado y relaciones que no pertenecen estrictamente a ninguno de los lenguajes de modelado que se va a componer.

3. Mediante el refinamiento de clases (*Class Refinement*)

Este tipo de composición se utiliza cuando uno de los lenguajes captura en detalle un concepto de modelado que existe solo como caja negra en un segundo lenguaje de modelado. Esta técnica se emplea de forma similar a la creación de interfaces, pero la relación entre los dos lenguajes de modelado compuestos viene dada por la contención jerárquica de los constructores de un metamodelo dentro de un constructor simple de otro metamodelo.

4. Mediante la instanciación de plantillas (*Template Instantiation*[6])

Esta técnica intenta superar una limitación que tienen las técnicas anteriores. Intenta dar soporte a la reutilización de patrones comunes de metamodelado o estilos en un metamodelo simple compuesto. Esta técnica consiste en registrar patrones comunes de metamodelado como plantillas de metamodelo abstractas. Luego, estas plantillas son instanciadas por lenguajes específicos de dominio.

4. Trabajos relacionados

Nuestra propuesta está relacionada con [10] en el sentido de que las dos trabajan con lenguajes específicos de dominio y tratan con la composición de los mismos a través de la composición de sus metamodelos. Sin embargo, la propuesta de Karsai et. al. no tiene un enfoque aspectual.

En el área de la orientación a aspectos hay una serie de propuestas relacionadas con la composición de modelos, como por ejemplo [3, 7, 9, 14].

En el ámbito del desarrollo del software orientado a aspectos, existen dos enfoques de composición, que se conocen como asimétrico y simétrico. En el enfoque asimétrico los aspectos y el modelo base o primario juegan distintos roles durante la composición, de tal forma que al modelo base se le van inyectando los aspectos. Este tipo de enfoque no suele aceptar composición entre aspectos, sino solamente entre aspectos y el modelo base. Un representante, a nivel de implementación de este tipo

de composición es AspectJ [18].

Por otra parte, en el enfoque simétrico, todos los elementos a componer son tratados de la misma forma, por lo que suele ser posible la composición de aspectos. Se puede decir que nuestra propuesta presenta un enfoque híbrido, ya que aunque se basa en un modelo base al que se le van inyectando aspectos, los aspectos se pueden componer, dependiendo de la plataforma en la que se vaya a implementar la aplicación.

5. Conclusiones y trabajos futuros

En este artículo se ha presentado una propuesta orientada a aspectos y dirigida por modelos para el desarrollo de aplicaciones web. Dentro de esta propuesta se ha realizado una exploración inicial de integración de aspectos a través de la composición de metamodelos.

Para lanzar esta propuesta es preciso crear una librería de aspectos, de tal forma que los diseñadores puedan disponer de un conjunto de lenguajes específicos de dominio para trabajar con los aspectos a un alto nivel de abstracción. Estos metamodelos se deberían poder extender y componer. Sin embargo, la extensión y la composición no deberían modificar los metamodelos de los aspectos originales.

Por otra parte, una de las cuestiones abiertas en investigación en la ingeniería basada en modelos es la especificación formal de la semántica de los lenguajes de modelado. Sería interesante estudiar que pueden aportar las ontologías a este tipo de problemática.

Referencias

- [1] Bhogill, P. S. *An Introduction to Java Server Faces*. Java News Brief. Aug, 2003. <http://www.ociweb.com/jnb/jnbAug2003.html>
- [2] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T. J. *Eclipse Modeling Framework: A Developer's Guide*. Addison-Wesley. 2003
- [3] . S. Clarke. *Extending standard UML with model composition semantics*. Science of Computer Programming, 44. 77–100. July 2002.
- [4] Colyer, A. *Towards Widespread Adoption of AOSD*. In R. Bodkin, A. Colyer, J. Memmert, A. Schmidmeier, editors, AOSD Workshop on Commercialization of AOSD Technology. March, 2003.
- [5] Eclipse Project. *Eclipse Modeling Framework*. Available at: <http://eclipse.org/emf>.
- [6] Emerson, M., Sztipanovits, J.. *Techniques for Metamodel Composition*. Proceedings of the 6th Workshop on Domain Specific Modeling held in conjunction with OOSPLA, pp. 123-139, October, 2006.
- [7] Gray, J., Bapty, T., Neema, S., Tuck, J. *Handling crosscutting constraints in domain-specific modeling*. Communications of the ACM. 44(10), 2001. ISSN: 0001-0782. pp., 87–93. ACM Press.
- [8] Greenfield, J., Short, K., Cook, S., Kent, S. *Software Factories. Assembling Applications with Patterns, Models, Frameworks and Tools*. Wiley Publishing, Inc. 2004
- [9] Grundy, J. *Multi-perspective specification, design and implementation of software components using aspects* International Journal of Software Engineering and Knowledge Engineering, Vol. 10, No. 6, December 2000, pp. 713-734.2000.
- [10] Karsai G., Maroti, M., Ledeczi, A., Gray, J., Sztipanovits, J. *Composition and Cloning in Modeling and Meta-modeling* IEEE Transactions on Control Systems Technologies, 12 (2). March, 2004.
- [11] Kiczales, G. and Hilsdale, E. and Hugunin, J. and Kersten, M. and Palm, J. and Griswold, W. G. *An overview of AspectJ*. Proc. ECOOP 2001, LNCS 2072. Editor J. L. Knudsen. Budapest. Springer-Verlag. Jun, 2001. pp. 327–353.
- [12] Lopes, C.V. D. *A Language Framework for Distributed Programming*, Phd. Thesis. College of Computer Science, Northeastern University. 1997. <http://www>.

parc.xerox.com/csl/groups/sda/pubs/papers/Lopes-Thesis/dissertation.pdf

- [13] *MDA Guide Version 1.0.1*, OMG/03-06-01. www.omg.org/mda.
- [14] Reddy, J. R., Ghosh, S., France, R. B., Straw, G., Bieman, J. M., McEachen, N., Song, E., Georg, G., *Directives for composing aspect-oriented design class models*, Transactions on Aspect-Oriented Software Development I, LNCS 3880, pp. 75-105, Springer-Verlag, 2006
- [15] Reina, A. M., Torres, J., Toro, M. *Towards developing generic solutions with aspects*. Proceedings of the Workshop in Aspect Oriented Modelling held in conjunction with the UML 2004 Conference. Lisboa, Portugal. Oct, 2004.
- [16] Special issue on Aspect-Oriented Programming. Communications of the ACM archive Volume 44 , Issue 10 (October 2001) ISSN:0001-0782. ACM Press.
- [17] AOSD Community Web page: <http://aosd.net>.
- [18] AspectJ Web page: <http://aspectj.org>