Thesauri managing and Software Agents: a proposed architecture

José Ramón Pérez Agüera
1Departamento de Sistemas Informáticos y Programación UCM.
jose.aguera@fdi.ucm.es

Rodrigo Sánchez Jiménez
Departamento de Biblioteconomía y Documentación UCM.
rsanchezj@ccinf.ucm.es

Abstract:

This article presents a basic proposal for the automation and use of thesauri for information retrieval in distributed environments through web services based on the Resource Description Framework (RDF) architecture. It begins by reviewing the proposals for descriptive tagging for thesauri coding that have appeared over the past four years. This is followed by a description of the basic architecture of a thesaurus implemented in Java. The article concludes by reviewing different communication and data exchange protocols, together with applications that can be used to implement this service. The text is accompanied by the computer application that has been developed.

Keywords:

Thesauri, thesaurus managing, software agents, semantic web, web services

1. Introduction

A thesaurus, as understood in classical information sciences like library sciences, is a combinatory tool consisting in lists of terms representing a determined technical or scientific domain and semantic relations linking those terms. These semantic relations cover three main types, equivalence, association and hierarchy which gives these kind of structures a flexible and highly descriptive behaviour. We consider these features as specially useful in a Information Retrieval techniques demanding environment like the Internet.

There are some internationally accepted directives aimed at guiding the construction of thesaurus, among which we can highlight ISO 2788:1986 and Z39.19:1993, its later evolution. According to these directives thesaurus are in fact linguistic tools aimed at providing IR capabilities to information systems, and although we can find some analogies with other tools like ontologies, the thesaurus structure is quite simpler and the difference between lexical and semantic levels is quite less steep than in the former case.

The use of thesaurus in specialised centres has been and still is very common in lots of specialised information centres. Anyhow modern automatic IR systems have not considered thesaurus as a way of semantic disambiguation or normalization of terms. Nevertheless some IR projects dealing with digitized linguistic resources have had great relevance during the 90's, including the world famous Wordnet. To a lesser extent thesaurus are also present in some of these projects which lead us to believe that using thesaurus in Information Retrieval is still possible and might yield interesting results.

Our first idea was to propose a way of designing an information service focused on acting as a middle layer between thesaurus and the client machines trying to access and operate with it. We saw this task as one among some others involved in the Information Retrieval process, and then decided to use some of the principles of distributed architectures in its creation, although maintaining our first intention of designing an independent web service. To cope with this task we first had to import the thesaurus to a machine readable format and implementing the basic access and query functionalities.

2. Thesaurus mark-up models

Before thinking of the implementation of thesauri server software a study of the main thesaurus codification initiatives was done. Among these initiatives we are to highlight those coming from the semantic web sphere. One of the facts we appreciated

was that almost all the thesaurus used till date had been marked up in some way, which made the idea of using them in a machine readable framework much more feasible. XML was clearly the candidate to mark up those thesauri, although there was still the need to decide from the various proposals based on it.

2.1. RDF/XML

The apparition of RDF is a turning point in the task of creating a semantic infrastructure to support a better use of information on the Internet. RDF is intended to represent resources in a way that makes them machine-readable and at the same time preserves their exact original meaning.

2.1.1 RDF and thesaurus codification

It did not take long for researchers to discover and begin to use RDF's capabilities to describe resources in the task of describing thesauri. From 2000 on various initiatives succeeded, although we will only talk in detail of CERES as the main precursor of these initiatives [1].

CERES was supported by the California Resources Agency [2] and was defined through the American standard of monolingual thesauri (Z39.19:1993). The goal of this initiative was to proportionate a method for thesauri exchange between applications. The reason behind the choice of RDF/XML was the natural capabilities of this language to express the various types of relations existing between concepts in a thesaurus.

Resources are treated as instances of Descriptor, Category or EntryTerm, which are at the same time subtypes of Term. This implementation uses typedNode's for every resource and a set of propertyTypes available for every Term, including scope notes, cataloguer notes, historical notes, source and status of the term, broader, narrower, related terms…etc.

RDF's application to thesauri is quite simple and functional which led to the adoption of this model by some important institutions [3]. Nevertheless now days none of these institutions still uses that model, which might be explained by the proliferation of W3C recommendations on knowledge organization systems and the general improvements achieved by mark-up technologies, all of which led to the adoption of newer more elaborated models.

2.2 OWL

The appearance of OWL, (Ontology Web Language) is to be considered a new turn point in knowledge organization oriented mark-up languages. OWL's creators inspired in the earlier developments of DAML-OIL, which gives OWL its descriptive logic flavour. OWL is a mark-up language intended to provide us with the capabilities of publishing ontologies on the web.

OWL was designed to be built over RDF and written in XML, although being able to represent ontologies with a much richer vocabulary and a stronger syntax than RDF [4]. This enables OWL to explicitly represent the meaning of terms from an existing vocabulary and expressing the relations existing between them [5].

OWL is divided in three sub-languages OWL-Lite, OWL-DL y OWL-Full, each of which provides us with an increasingly complex work set, from the simplicity of OWL-lite to the full-fledged ontology capabilities of OWL.-Full. OWL-lite provides us with the basic features necessary to code ontologies and many other simpler knowledge systems, and represents as the W3C states, the fastest way of migrating thesaurus and other taxonomies to a semantic web environment [6] .

As well as in the case of RDF we can find some concrete proposals leading to the use of OWL in the task of representing thesauri. We can highlight D.H. Fischer's on the representation of the German National Cancer Institute, available on the Internet [7]. One of the most interesting points of these proposals is the use of OWL-Lite as a way of converting a thesaurus in an ontology, which leads Fischer to plan the very structure of these linguistic tools.

2.3 SKOS-Core

Now days we can find a very specific proposal form thesauri representation in the framework of the semantic web, SKOS-Core. SKOS-Core is an rdf schema intended to represent thesauri and other similar knowledge organization systems. This poses the problem in the same terms that CERES did, although this proposal apart from being developed in the environment of W3C's activities provides us with more elaborated mechanisms.

The main goal of SKOS-Core is to provide a way of migrating knowledge organization systems to the field of the semantic web. Apart from this it also allows to build up simple conceptual schemes to be used on the web. SKOS-Core is intended to be a complement to OWL, providing us a basic framework for making concept schemes but without the semantic strictness imposed by OWL. We can talk of it in terms of a further simplification of OWL-Lite, which on the other hand provides easier access to a wide number of people to this kind of knowledge organization systems.

SKOS-Core is thus a simple model of easy use and fast learning ready for a greater number of people,

which can help on generalised semantic web expansion in contrast with the current isle-like spreading model [8].

The basic idea behind this RDF schema is to provide a way of expressing concepts and concept schemes. A concept might be thought of as a unit of thinking that can be defined or described. At the same time a concept scheme is plainly a collection of concepts. A concept might have a series of associated tags, where every tag is a word sentence or symbol used to refer to that concept.

Each concept can only have a single preferred tag, which is known as a descriptor or preferred term, and an unlimited number of alternative tags known as non-descriptors or non-preferred terms, a common use terminology in the field of controlled vocabularies.

Apart from this we can find the usual thesaurus kind of relations, that is, equivalence, hierarchy and association all of which can be assigned to concepts belonging to a same concept scheme, being a concept scheme the equivalent of a thesaurus with relations being non syntactic but semantic ones. At the same time we can map the equivalences of terms belonging to different thesauri, a very useful feature for knowledge exchange.

With SKOS-Core we can identify every concept using an URI, which gives it independence over any specific thesaurus. Afterwards we can associate multiple non-preferred terms and a preferred term to connect with the concept. Scope notes are also easy to include, and can provide a comprehensive description of the concept.

Apart from defining concepts SKOS provides capabilities to define semantic relations, like hierarchic and associative ones. The properties skos:related, skos:narrower and skos:broader are grouped around the property skos:semanticRelation. The last thesaurus typical relation, the equivalence one, is already considered as an alternative term, that is, the skos:altLabel property.

SKOS-Core does also provide ways of representing concepts in other languages through the use of a special multilingual mark-up, which allows for duplication of both preferred and not preferred term labels as many times as necessary. All these properties provide us with the means to describe almost every thesaurus with ease, but SKOS provides some more instruments to accurately defining the previous semantic relations.

skos:broaderGeneric and skos:narrowerGeneric can specify inclusive relations between concepts. To define a concept as an instance of another skos:broaderInstantive and skos:narrowerInstantive can be used. To express a concept being part of others skos:broaderPartitive and skos:narrowerPartitive will suffice to further develop the idea of subordination. skos:relatedHasPart and skos:relatedPartOf on the other hand express more precise associative relations.

To end with this review of the W3C proposal on thesauri we should say that SKOS-Core does also allow defining top concepts or hierarchy heads through the skos:TopConcept property which can be use to establish facets for a given thesaurus.

2.4 Other mark-up models for thesauri representation on the Internet.

Although the main goal of this paper is to set thesauri automatization in the context of the semantic web we can't avoid briefly mentioning other relevant thesauri automatization and knowledge organization initiatives.

2.4.1 Zthes

Zthes describes an abstract model for the representation and search of thesauri just like the ISO 2788 norm prescribes [9]. The basic idea behind this proposal is that of planning a model that enables the implementation of thesauri in a way that permits access through the Z39.50 and SRW protocols. This proposal, given the peak of mark-up technologies, does also offer a DTD for thesaurus representation, although it is mainly a Z39.50 focused initiative.

2.4.2 Topic Maps

We have a second proposal derived from topic maps, which is a standard for conceptual browsing that enables the representation of thesauri. This quite a venerable proposal, first developed during the times of SGML and HyTime, although it has lately been renewed through the adoption of a DTD for Topic Map representation [10].

2.4.3 SKOS-Core vs. Zthes y Topic Maps

Both Topic Maps and Zthes come from old development lines, which has impelled them to adapt to the XML trends lately. Nevertheless these are perfectly functional technologies, although are somehow limited to work in narrower application fields, which contrasts with the standardization prospective of the W3C initiatives. This is the main reason that made us choose an RDF/OWL/SKOS-Core line of work.

3. A Thesauri managing agent architecture

Once we have chosen SKOS-Core to code our thesauri we can describe the design and implementation of the thesauri server core. The core

of the system is composed by two main classes due to manage the basic functionalities to access the thesaurus [11], “Thesaurus” and “Concept”, and two more due to give us basic conceptual functionalities, “NormalizarTermino” (Normalize Term) and “Searcher”.

The Thesaurus class is designed to abstractly represent a thesaurus modelling it to be instantiated and used by every other class needing it. The Thesaurus class contains a TreeMap as a class attribute ready to store data relating to the thesaurus, where the key / value couple is defined by descriptors and non-descriptors (keys) that act as entries to the thesaurus, and Concept objects which refer to the concepts behind the surface level (thus being values of those keys). Each preferred or non-preferred key contains a link to the value, a given Concept object.

As we said before the Thesaurus class uses Concept type objects to abstract the concept of descriptor. This Concept object includes attributes related to the components belonging to a given descriptor in traditional thesauri. This way we can find a String [12] related to the preferred term that usually represents the descriptor and thus its underlying concept. At the same time two more String variables are use to translate the term to French and Spanish. A further String holds scope notes about meaning and use of the current concept during indexing. Finally a series of lists implemented through ArrayList, and referring to related terms, broader, narrower and non-descriptors are also part of the Object.

Through this structure we can define a concept in a very similar way to other digitalized linguistic resources used in Information Retrieval and Natural Language Processing during the last 10 years or so, representing it through semantic relations with other concepts, although we still maintain the basic features required by the ISO 2788 standards.

Both classes implement the Serializable interface, which is related to the fact that both the thesaurus and its descriptors are stored in the same file, where it can wait for further usage after runtime is finished. Every time the application is initialized those classes are loaded in memory to be used during execution. This does also allow for multiple simultaneous thesauri loading, resulting in cross thesauri concept-mapping capabilities. This implementation does not excessively affect efficiency and allows us to maintain an object oriented (developer-friendly) focus on the question of design and implementation.

Given this core we have designed a series of classes to illustrate the possible operations to be executed on a thesaurus. The class NormalizarTermino lets us automatically normalize concepts against the thesaurus. Having an entry composed of a single word or set of words this class is able to return the terms normalizing concepts represented by the submitted term or terms. This process is done over the complete thesaurus using preferred and non-preferred terms as entry-points. This way we can guarantee the coherence of the normalization process with respect to the thesaurus we are using. At the same time it allows us to retrieve related terms for every descriptor, which might be used in query expansion tasks.

The Searcher class retrieves query results in a RDF/SKOS-Core format with the full set of information available for the current term. One of this query-response examples is given bellow:

jose@leviathan:/eclipse/workspace/ThesaurusAgent$ java thes.Searcher cáncer

* ** *** ****

<rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”

xmlns:skos=”http://www.w3.org/2004/02/skos/core#” >

<skos:Concept rdf:about=”http://spines/neoplasmas%20malignos”>

<skos:broader rdf:resource=”http://spines/enfermedades”/>

<skos:related rdf:resource=”http://spines/transformación%20neoplásica%20celular”/>

<skos:prefLabel>neoplasmas malignos</skos:prefLabel>

<skos:prefLabel xml:lang=”en”>malignant neoplasms</skos:prefLabel>

<skos:prefLabel xml:lang=”fr”>neoplasmes malins</skos:prefLabel>

<skos:related rdf:resource=”http://spines/neoplasmas%20benignos”/>

<skos:related rdf:resource=”http://spines/hábito%20de%20fumar”/>

<skos:related rdf:resource=”http://spines/enfermedades%20incurables”/>

<skos:altLabel>cáncer</skos:altLabel>

<skos:altLabel>carcinoma</skos:altLabel>

<skos:related
rdf:resource="http://spines/i+d%20médica"/>

<skos:related
rdf:resource="http://spines/neoplasmas%20experimentales"/>

<skos:related rdf:resource="http://spines/pechos"/>

<skos:narrower
rdf:resource="http://spines/neoplasmas%20inducidos%20por%20radiación"/>

<skos:related
rdf:resource="http://spines/antineoplásicos"/>

<skos:related
rdf:resource="http://spines/enfermedades%20de%20la%20mama"/>

<skos:related
rdf:resource="http://spines/enfermedades%20gastrointestinales"/>

<skos:broader
rdf:resource="http://spines/neoplasmas"/>

<skos:related
rdf:resource="http://spines/condiciones%20precancerosas"/>

<skos:related
rdf:resource="http://spines/enfermedades%20ginecológicas"/>

<skos:related
rdf:resource="http://spines/cancerígenos%20ambientales"/>

<skos:related
rdf:resource="http://spines/enfermedades%20del%20aparato%20genital"/>

<skos:related rdf:resource="http://spines/amianto"/>

<skos:narrower
rdf:resource="http://spines/leucemias"/>

<skos:narrower
rdf:resource="http://spines/sarcoma"/>

</skos:Concept>

</rdf:RDF>

As we can see every matching query is answered with a RDF document defined by the SKOS-Core schema referring to the corresponding concept [13].

The main target of both classes is to show some working examples of the use of the functionalities offered by Thesaurus and Concept, both for human and machine queries, in the case of normalization or directly for the use of automatic indexing systems, in the case of Searcher. Using this basic description of the classes we can define a basic core for a thesauri server implementing both the knowledge structure (the thesaurus) and the operations to be executed on it.

3.1 Inner functionalities of the Thesauri server

As we have stated elsewhere our view of a thesauri server goes far beyond the use of the thesaurus and also deals with maintenance and actualization. The need for thesauri actualization is a problem that has been observed in the Librarian Sciences literature. Blanca Gil highlights the capabilities of the thesauri structure, which might be enlarged and modified according to the needs derived from its continuous use [14]. According to this we tried to develop a system in which automatic actualization of terminology was possible, using transactional analysis of user's queries or any other means.

The idea behind all this is the possibility of creating intelligent agents that learn user's preferences referring to terminology, as well as being capable of processing thesauri preferred and non-preferred terms as queries to be posed to any collection and evaluated to check its retrieval capabilities with the final goal of automatically performing thesaurus actualization.

4. Communication with other applications

The thesauri server enables us to use thesauri in the environment of both manual an automatic indexing and as a hole in any Information Retrieval system. Having said this we still have the need of adopting a standard to enable the use of the thesaurus by other applications in IR distributed systems.

In the next section we briefly describe some of the various remote access formulas to be used by a thesauri server. We will discuss the options to be taken in order to provide the server with both flexibility and wide communication capabilities.

4.1 Communicating using FIPA-RDF

The software agents' paradigm provides us with a solid background in which we can base communications between informatic applications. Because of this the first thing to do is weighting the possibility of considering our thesaurus server as an agent. According to this we can consider the system as

an agent providing services relating to term normalization to be used by other IR applications such as search engines, automatic indexers…etc.

We can introduce one of these external applications as another agent, an automatic indexing agent. This agent would need to check the proper form of terms before statistically or otherwise treating words in a document collection. To achieve this it should communicate with the thesaurus managing agent so It can retrieve the needed normalizations. Thus both agents communicate and cooperate to cope with a singular problem (conceptual normalization of index terms).

The FIPA standard divides communication between agents in "communication events", "interaction protocols" and "content languages". Communication events are composed of the blocks in which a dialog between two applications can be divided in. They define the meaning of messages apart from any particular context. The interaction protocols define a message sequence representing the complete dialog between agents. Finally content languages establish the language to be used for the message content.

Our thesauri server, which we will treat as a thesauri managing agent in this section, can adopt this way of communicating with other applications, thus being able to share both contents and operations' results. Using a FIPA-RDF message sequence we can establish a coordinated dialog between two or more agents, thus setting the thesauri management agent as the centre of a multi-agent system. Further more, this way of communicating allows us to directly use RDF without the need of transforming the original RDF documents, which perfectly matches the chosen thesauri mark-up technology (SKOS-Core).

4.2 Conceptual normalization web services using soap

There is another way of posing the problem of communication between applications, apart from the ones derived from the software agents' paradigm. In the last years we have seen the uprising of web services closely bounded to the development of mark-up languages. Namely XML based web services have proved to be a feasible approach to the problem of cross-application information sharing and function invoking, going beyond Operative Systems and platforms. Although XML web services are usually independent from each other, we can find a way of bounding them together so that they cooperate in a determined task.

Web services are not intended to substitute common applications in most of heavy-duty or medium-size operations, but constitute an interesting alternative to solve small-sized operations, such as exchange of information or request of minor information processing, such as retrieving the ISBN number of a given title of book, retrieving results of a query…etc. We can find a very good example of these applications in Google's SOAP based web service, which facilitates the use of it's search engine for various tasks [15].

In our case we will use web services to let various external applications access to the thesaurus normalization and query functions already described. The main idea behind this is to use these functionalities without having to worry about the inner workings of the system, a "black box approach" if you wish.

To implement such a service we can use SOAP (Simple Object Access Protocol) a protocol proposed by the W3C based upon XML and designed to make structured information exchange between applications easy. SOAP does permit (as well as FIPA-RDF) RDF code embedding, so that we can include it in the messages' body [16]. This way we can communicate with a different array of applications without having to change anything of the original thesauri server responses.

5. Conclusions

As we can see it is possible to give our thesauri managing application very good communicating capabilities. The target of the various query interfaces presented before is mainly that of widening the number of applications able to use thesauri as a knowledge base for various IR tasks (or any others deemed convenient), given the fact that thesauri are a very common and standardized way of coding knowledge.

Notes

[1] We can find an excellent review of the various thesaurus mark-up proposals in http://www.w3c.rl.ac.uk/SWAD/deliverables/8.2.html#4.1

[2] Available in http://ceres.ca.gov/thesaurus/RDF.html

[3] http://ceres.ca.gov/thesaurus/

[4] http://wwvw.w3schools.com/rdf/rdf_owl.asp

[5] http://www.niso.org/standards/standarddetail.cfm?stdid=518

[6] http://www.w3.org/TR/2004/REC-owl-features-20040210/

[7] http://www.mindswap.org/2003/CancerOntology/

[8] Eva M. Méndez Rodríguez (2002) Metadatos y recuperación de información: estándares, problemas y aplicabilidad en bibliotecas digitales, Gijón, Trea

[9] http://zthes.z3950.org/

[10] http://www.topicmaps.org/

[11] The word class comes from the terminology used in object oriented programming. The application was implemented using Java, what explains the extensive use of object oriented terminology throughout this text.

[12] A String is a data type referring to any set of characters.

[13] RDF generation was done using Jena 2.1

[14] Gil Urdiciain, Blanca (1996) Manual de lenguajes documentales, Madrid, Noesis, p. 215–220.

[15] http://www.google.com/apis/.

[16] Ogbuji, Uche, Using RDF with SOAP: beyond remote procedure calls, http://www-106.ibm.com/developerworks/webservices/library/ws-soaprdf/