

USING A LARGE LINGUISTIC ONTOLOGY FOR INTERNET-BASED RETRIEVAL OF OBJECT-ORIENTED COMPONENTS

Stefano Borgo⁺, Nicola Guarino⁺, Claudio Masolo^{*}, Guido Vetere⁺⁺

(⁺) National Research Council, LADSEB-CNR, Padova

(^{*}) University of Padova, Dep. of Electronics and Computer Science

(⁺⁺) Consorzio CORINTO, Bari

1. INTRODUCTION

High-quality software retrieval is emerging today as a key problem for the whole software market, both on the side of software industry (software reuse) and on that of end-users (software use). The availability of world-wide sources of downloadable software and the success of object-oriented technology make this problem more evident: object-oriented components are supposed to be “intrinsically” reusable, but the costs of *locating* those which satisfy a given description, *assessing* their relevance, and *adapting* them to the current needs are still high. In a recent survey paper, Mili *et al.* [Mili *et al.* 1995] analyze such costs in detail, observing that the costs of relevance assessment can be kept low by increasing the expressiveness of the language used for encoding component properties and formulating queries, enhancing therefore the quality (*precision*) of the retrieval.

Faceted classification schemes like those proposed in [Prieto-Diaz 1991] can be used to enhance such expressiveness, possibly using taxonomies of keywords in order to alleviate (but not eliminate) for the user the burden of sticking to a limited vocabulary.

More recently, some projects focused on software reuse have adopted a rather expressive language based on *description logics* [Devanbu *et al.* 1991,1994], in order to make possible both detailed specifications of software functionality and structure, and sophisticated queries. Moreover, such systems can exploit the *dynamic classification* mechanism offered by description logics, which relieves the user from the burden of depending on a rigid taxonomy.

However, as underlined by Mili *et al.*, the expressiveness of the language adopted for component encoding “is limited by the developer’s willingness to formulate long and precise queries”: if typical queries are relatively simple, then detailed descriptions of component’s structure become useless from the retrieval point of view. This seems to be especially true in the case of object-oriented software, which is typically specified in terms of its functional properties rather than in terms of its structure. The focus on functional description is also justified within our perspective of heterogeneous software distributed over the

Internet, very different from the “in-house” scenario typical of other software reuse projects like LaSSIE.

The system described in this paper adopts a language of limited expressiveness, privileging the simplicity of use as the most important requirement. We adopt a very simple graph structure for representing both queries and component data, but – differently from most of current systems – we do not assume the user to have familiarity with the vocabulary used for component encoding, relying on a large *linguistic ontology* like Sensus [Swartout *et al.* 1996] to perform the match between queries and data. In the encoding phase (which we assume to be a *manual* process supported by an interactive environment), a software analyst describes a component by a simple graph where nodes and arcs are labelled with English words. Since binary relations are not usually denoted by nouns, a special semantics is adopted for this graph, which is called *Lexical Semantic Graph*. English nouns appearing in the graph are recognized by a lexical interface based on Wordnet [Miller 1995], which asks the analyst to choose among possibly different senses associated to each word. The graph of words is therefore translated into a graph of senses, each one corresponding to a node in the Sensus ontology. The query graph is built by the user in a similar way, but the words chosen and the corresponding senses can be of course different, as well as the structure of the graph. Conceptually, the search process implements a *graph matching* algorithm, returning the identifiers of all components whose description is subsumed by the query.

In the following section, we describe the main design choices of a project on software retrieval currently going on at Corinto¹, a research consortium established to study and promote object-oriented technology. In section 3 we present the encoding and retrieval process in some detail, with the help of a concrete example. Finally, in section 4, we briefly discuss the advantages and drawbacks of this approach.

¹ Consorzio di Ricerca Nazionale Tecnologia Oggetti, as partnership of IBM Semea, Apple Italia and Selfin SpA.

2. MAIN DESIGN CHOICES

While addressing the task of Internet-based software retrieval in the perspective outlined above, one of the practical requirements we recognized first was that — at the expense of a greater initial effort — the technology to be developed had to be able to apply to a variety of information sources. Since an early stage, the project has been given therefore a very general perspective, and software components have been simply considered as information documents to be described and retrieved. In other words, our approach to software retrieval can be seen as a particular case of ontology-based information retrieval. The availability of large linguistic ontologies which can be readily imported and used¹ has motivated the choice of avoiding the construction of an *ad-hoc* ontology built on a limited set of primitive keywords, relying rather on the power of a full English vocabulary. This is the key presupposition of this project, which can be seen itself as an experiment of large-scale ontology reuse. Of course, some modifications to this ontology will be necessary, and indeed the study of the lessons learned in this adaptation process will be one of the side-effects of this experiment.

On the basis of these considerations, we decided to implement OntoSeek, a general purpose tool for ontology-based information retrieval², provided with very few inference capabilities but fast and reliable enough to enhance or even replace keyword-based information retrieval in many cases. The system has been designed to:

- encode an information token by means of a simple graph of concepts and relations tagged by English words;
- query a database of (previously encoded) information tokens by using possibly different words with respect to those used in the encoding process;
- browse ontologies of word senses

Moreover, with the aim of crafting a modern industrial-strength tool enabling the management of information repositories based on shared ontologies, a number of high priority requirements have been assumed:

- efficiency on data volumes
- system scalability/portability
- state of the art Internet capabilities

2.1 BASIC ARCHITECTURE

The basic architecture of the system is sketched in Fig. 1. We summarize here the role of the main modules, postponing a more detailed description to the example discussed in the next section.

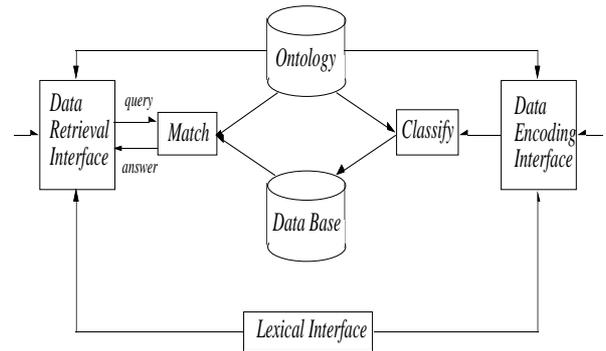


Fig. 1. Basic architecture of OntoSeek.

The software analyst draws a *Lexical Semantic Graph* (LSG) (e.g., the one showed in Fig. 2), representing the functional description of a given component. As discussed below, the main feature of this graph is the fact that its components are denoted by lexical items. This graph is then interactively refined into a *Sense Graph* by the *Data Encoding Interface*. The knowledge about possible word senses is given by the *Lexical Interface*³, while all the semantic knowledge is contained in the Sensus ontology. The sense graph is the input of a *Classification* phase, where the graph describing the component is added to the *Database*, suitably linked to the *Ontology* in order to exploit efficient search algorithms. The data retrieval process works roughly in a dual way: the query is represented again as a word graph, which is then refined into a sense graph within the *Data Retrieval Interface*. The database is then searched in order to find the components described by those sense graphs which are subsumed by the query, according to the taxonomic constraints present in the ontology.

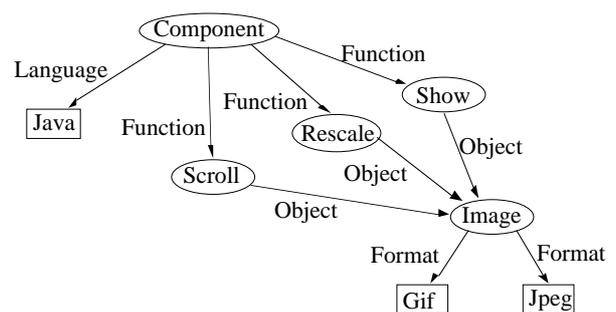


Fig. 2. A Lexical Semantic Graph representing a software component.

¹ Like Pangloss and Wordnet, recently merged together – with other linguistic sources – to build Sensus.

² More exactly, *lexicon-driven information retrieval*.

³ Currently, we are planning to use Wordnet for this purpose. Notice that Wordnet also includes a taxonomy of word senses, which in the figure above is assumed to be part of the ontology.

2.2 LEXICAL SEMANTIC GRAPHS.

Let us now discuss the semantics of the graph reported in Fig. 2. The basic idea of the whole project is to exploit readily available lexical knowledge in order to describe and retrieve information items. Now most of the labels currently used to denote binary relations (like “part-of”, “function-of”, “to-the-right-of”...) do not correspond to lexical entries in available electronic thesauri like Wordnet. In fact, people are usually forced to invent ad-hoc relation labels when using current conceptual modelling formalisms like ER, OMT, conceptual graphs or description logics. Many of these labels (like “function-of”, “has-part”...) are however formed from standard nouns; such nouns are indeed called “relational nouns” since they have a direct relational import, in the sense that their meaning can be fully understood only in the context of a binary relation: the noun “part” denotes the property of being a part (of something not specified), but also the range of the relation “has-part”. This situation has been discussed in [Guarino 1992], where a relation like “has-part” is defined as the *relational interpretation* of the noun “part”. According to Guarino, only relations of this kind (i.e., relational interpretations of concepts) should be modeled as attributes (also called “roles”, or “slots”) of objects; the remaining relations should be rather modelled as constraints *between* objects.

In the light of this discussion, we can assume that arcs labelled with nouns denote the relational interpretations of such nouns, while arcs labelled with (transitive) verbs denote the relations corresponding to such verbs. In this way we are able to always guarantee a *lexical handle* to interpret the arcs appearing in our graph¹. More formally, we call *Lexical Semantic Graph* a graph like the one above, where (see Fig. 3):

1. ellipses denote concepts and rectangles denote individuals;
2. ellipses and arcs are labelled with nouns or verbs²;
3. if a verb appears as a label of an ellipsis, then it denotes the concept corresponding to the nominalization of that verb.
4. if a transitive verb appears as a label of an arc, it denotes the relation existing between the subject and the object of that verb.
5. if a noun C appears as a label of an arc it denotes a relation whose range is restricted to the concept C (i.e., the relational interpretation of C). In particular, an arc with label C from the concept (or individual) A to the concept (or individual) B denotes a non-empty relation

¹ It is interesting to note that the opportunity of a linguistic restriction on role names has been advocated in a historical paper by Bill Woods [Woods 1975], who proposed a “linguistic test” which can be paraphrased as follows: A is a good role-name for X if for each filler F we can say that *F is an A of X*. See [Guarino 1992] for a full discussion.

² Infinitive forms are supposed to be abbreviations for the corresponding nominalizations: “show” denotes the concept of “showing” (whose instances are events).

with domain A (or {A}) and range $C \cap B$ (or $C \cap \{B\}$).

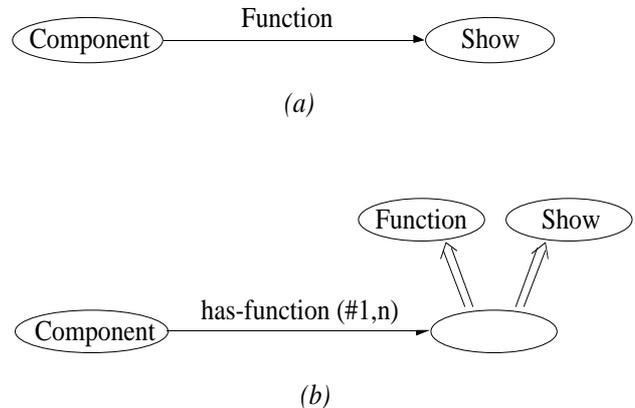


Fig. 3. Semantics of Lexical Semantic Graphs (LSG). The LSG in (a) is a shorthand for the ordinary semantic graph reported in (b). The advantage is that the relation “has-function” is represented by means of the lexical item “function”.

From the point of view of expressiveness, an LSG is equivalent to the description logic FL^+ (known to be polynomial), with the trivial addition of the role-forming operator *range*. For instance, the graph in Fig. 3a corresponds to (and Component (some (range Function))) (all (range Function) Show)³.

2.3 MAIN IMPLEMENTATION CHOICES

As for the implementation of the ontology is concerned, a C++ data structure has been preferred to the more traditional LISP-based technology.

Such a data structure, a C++ class named “ontology”, is mainly implemented as an extension of a template ANSI-C++ class “graph” providing an internal structure and a set of methods designed to manage graphs of objects into the system persistent memory. Object persistency is achieved by an Object-Oriented DBMS (ODI ObjectStore™). *Vertices* and *arcs* of “ontology” are tailored to represent *classes*, *metaclasses*, *parameters*, *individuals* and a variety of relations among them. Each individual can be associated to one or many Internet URLs representing document pointers.

The user interface is entirely written in Java. Once running on a Internet client, it communicates with the OntoSeek server by means of a low-level efficient protocol. The interface enables the graphical browsing of the ontology, the selection of relevant concepts and the visual composition of both queries and data descriptions.

As a result of its implementation, the system is expected to efficiently manage very large ontologies. A complete discussion on system efficiency is out of the scope of this paper. We just mention here that all the “ontology”

³ See for instance [Woods and Schmolze 1992] for a review of description logics.

methods are based on basic graph algorithms whose complexity is discussed in [Horowitz and Saini 1983].

OntoSeek accepts a subset of Ontolingua as an input format for the ontology, where slots denoted by a particular syntax are given a semantic related to the lexical component. A translator has been developed in order to transform Ontolingua statements into the internal data representation.

3. A CONCRETE EXAMPLE

Let us now present the main steps of the encoding and retrieval process, with the help of a concrete example.

3.1. ENCODING.

Let us suppose that input information about the component to be encoded is given by a simple natural language description like the following one:

JAVA component allowing the visualization of GIF and JPEG images, supporting image scrolling and rescaling.

1. With the help of the data-entry interface, a software analyst (supposed to understand the meaning of the above sentence, and possibly acquainted with the component to be encoded) comes up with the LSG reported in Fig. 2, trying to account for all the relevant information at the most specific level of detail. At the time being, its only requirement is that the labels attached to arcs and concepts are correct English words. Of course, the mental process behind this step is not obvious at all, and suitable tools need to be devised in order to extract the relevant information. For instance, in the statement above, the verbs “allowing” and “supporting” have been ignored, focusing the analysis of the main functionalities of the software. If the characteristics of the data items to be encoded are homogeneous, it is possible in this phase to facilitate the data-entry activity by proposing the user a *template graph* to be refined, where for instance usual *facets* like “Function” and “Object” appear as arc labels.
2. The English labels appearing in the LSG need now to be disambiguated, in order to isolate their specific *senses*. This operation is performed interactively with the help of the Wordnet lexical interface, which proposes to the user different senses (concepts) for each word, together with a brief textual description. Disambiguation decisions are also supported by the ontology, since it is possible to visualize the taxonomic relationships between concepts, and, if present, further semantic constraints. For instance, Wordnet presents the following possible senses for “Function”, among which the user selects Sense 2:

Sense 1
function, mathematical function -- (a mathematical relation such that each element of one set is associated with at least one element of another set)

=> mathematical relation -- (a relation between mathematical expressions (such as equality or inequality))

Sense 2
function, purpose, role, use -- (what something is used for; "the function of an auger is to bore holes"; "ballet is beautiful but what use is it?")
=> utility -- (the quality of practical use or usefulness)

Sense 3
function, office, part, role -- (the actions and activities assigned to or required or expected of a person or group: "the function of a teacher"; "the government must do its part" or "play its role" or "do its duty")
=> duty -- (work that you are obliged to perform for moral or legal reasons: "the duties of the job")

Sense 4
function -- (a formal or official social gathering or ceremony; "it was a black-tie function")
=> social gathering -- (a gathering for the purpose of promoting fellowship)

Sense 5
routine, subroutine, subprogram, procedure, function -- (a set sequence of steps, part of larger computer program)
=> software, software system -- (written programs or procedures or rules and associated documentation pertaining to the operation of a computer system)

Notice that, browsing the ontology, the user can decide to modify the concepts (senses) associated to the graph in order to refine the meaning expressed by the original labels. This can be especially useful if the original label is too generic, and the ontology allows for a more specific concept. The result of this step is as LSG labelled with *concepts* rather than *words*.

3. With the help of the ontology, the concept graph produced in the previous step is now checked for semantic consistency. This step is very much dependent on the richness of the knowledge available in the ontology. In the current phase of the implementation, concept graphs are assumed to be already semantically valid.
4. The concept graph is now stored in a permanent database, with suitable links to the ontology that make possible its efficient retrieval in response to a more generic query.

3.2. RETRIEVAL

The first step of the retrieval phase are conceptually very similar to the encoding phase. Let us suppose that the user formulates a query like “component suitable to represent and resize pictures”.

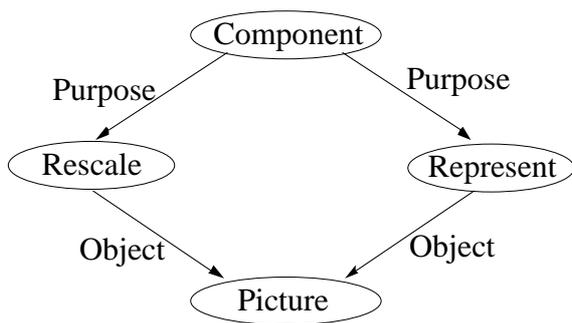


Figure 4. A query graph subsuming the graph of Fig. 2.

1. The user enters the LSG reported in Fig. 4 above.
2. The English labels appearing in the LSG of Fig. 4 are disambiguated with the help of the Ontology and the lexical interface. This phase may be simplified if the context is fixed, since certain classes of senses may be excluded a priori.
3. The resulting concept graph is used to search for information items described by more specific graph. In our case, the graph of Fig. 4 subsumes that of Fig. 2, since in Wordnet there is a common sense for “purpose” and “function” and a common sense for “picture” and “image”, while “resize” is more general than “rescale”, and “represent” is more general than “show”.

4. DISCUSSION

Let us now discuss the approach presented above in the light of the current literature, with special regard to systems based on faceted classification schemes. With respect to these approaches, our main points are:

- a simple (but semantically rigorous) representation language of intermediate expressivity (roughly between facets and description logics);
- a very large choice of description terms both in the encoding and retrieval phase, due to the use of a rich English lexicon combined with a large ontology;
- the possibility of semantic checks guided by the ontology;
- an efficient Internet-based search engine.

Regarding facets-based schemes, a well known limitation is related with the fixed number of descriptors (keywords). When we consider a large repository of non-homogeneous data, the choice of such descriptors can be very difficult. Classifying these data in a satisfactory way implies defining a big set of descriptors, whose exact meaning is difficult to control, document and maintain. It may even happen that the intended meaning of a descriptor varies depending on data types. When dealing with homogeneous data, the importance of this problem decreases, but it does not disappear. Suppose we have a repository of products of similar kind (like payroll programs, for instance). A few

facets may be enough to describe it, but the keywords to be allowed under each facets may be quite many.

In the lexicon-driven approach we have described these problems are overcome. The classification is flexible enough to describe large non-homogeneous repositories, where it is necessary to represent a variety of coarse distinctions, as well as homogeneous repositories, where fine-grained distinctions are important. Moreover, the ontology (supported by the lexical interface) makes possible to clarify the meaning of each term used.

In sum, the real importance of facets-based classification systems lies in the particular structure of keywords they propose, designed to be effective for the information retrieval task. Within our approach we can still use this structure as a *template*, offering however to the user three extra-services: the possibility to *paraphrase* the description, the possibility to *add meaning* to the description, and the possibility to *better understand* the meaning of the words adopted. In this perspective, a lexicon-driven system can be seen as a smooth but powerful improvement with respect to a facets-based system.

Regarding the system as an Internet information retrieval service, OntoSeek is characterized by an high interactivity, which relies on Java language, an efficient server implementation and a low-level communication protocol. These features allow users to effectively perform the complex dialogue leading to query formulation which is at the basis of our approach.

Of course, the system we have discussed has many limitations, which need to be taken into due account. As an example, consider an item called “RAM doubler” which does not magically double the physical RAM (unfortunately), but does enrich the performance of a computer. If the software analyst uses the very same words, “Ram” and “double”, to classify this software, then a query like “enhancing performance”, would never match the above object. Note that even in the case where the item would be described by the expression “enhancing memory”, the query could not be satisfied, unless knowing that the performance of a computer can be enhanced by increasing the performance of a *part* of it, namely its memory. This kind of problem is faced in each approach based on just natural language descriptions, where no background domain knowledge is available.

ACKNOWLEDGEMENTS

This work has been done in the framework of a cooperation between LADSEB-CNR and CORINTO. We are grateful to Gennaro Laera for clarifying some implementation aspects of the project.

REFERENCES

- Devanbu, P., Brachman, R., Selfridge P., and Ballard, B. 1991. LaSSIE - A Knowledge-Based Software Information System, *Communications of the ACM*, 34(5)
- Devanbu, P and Jones, M. 1994. The Use of Description Logics in KBSE Systems. In *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy.

- Guarino, N. 1992. Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. *Data & Knowledge Engineering*, 8: 249-261.
- Horowitz, E. and Saini, S. 1983. *Fundamentals of Data Structures*. Computer Science Press.
- Mili, H., Mili, F., and Mili, A. 1995. Reusing Software: Issues and Research Directions. *IEEE Transactions on Software Engineering*, 21(6): 528-560.
- Miller, G. A. 1995. WORDNET: A Lexical Database for English. *Communications of ACM*(11): 39-41.
- Prieto-Diaz, N. 1991. Implementing Faceted Classification for Software Reuse. *Communications of the ACM*, 34(5): 88-97.
- Swartout, B., Patil, R., Knight, K., and Russ, T. 1996. Toward distributed use of large-scale ontologies. In *Proceedings of 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*. Banff, Canada.
- Woods, W. A. 1975. What's in a Link: Foundations for Semantic Networks. In D. G. Bobrow and A. M. Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*. Academic Press: 35-82.
- Woods, W. A. and Schmolze, J. G. 1992. The KL-ONE family. In F. W. Lehmann (ed.) *Semantic Networks in Artificial Intelligence*. Pergamon Press, Oxford: 133-177.