An Architecture for a Federation of Highly Heterogeneous Lexical Information Sources

Artola, X.
Dept. of Computer Languages and Systems
University of The Basque Country
649 p.k., 20080 Donostia (Basque Country)
jiparzux@si.ehu.es

Soroa, A.
Dept. of Comp. Science and Artificial Intelligence
University of The Basque Country
649 p.k., 20080 Donostia (Basque Country)
ccpsoeta@si.ehu.es

Abstract

The purpose of this paper is to present an architecture for a federation of highly heterogeneous lexical information sources. We address the problem of querying very different existing sources of lexical information from the point of view of information integration, using for this purpose a unique and common query language. The so-called local-as-view paradigm is used for describing each lexical source as a view over a general conceptual model. A general conceptual model for describing lexical knowledge is presented, as well as the way to describe each source in terms of the classes and relationships of the general model. Both the conceptual model and the sources are specified and implemented using a description logic language, and an algorithm that translates queries from the general model into each particular schema has also been implemented. Finally, an example of how a query is processed and some conclusions are given.

Introduction

The purpose of this paper is to present an integrating architecture for a federation of highly heterogeneous lexical information sources. The term *lexical information source* is used here in its broadest sense, and so it includes from very different lexical and dictionary databases, up to heterogeneously structured and coded electronic dictionaries or language-processing lexica, or even language-processing programs such as lemmatisers or part-of-speech taggers. Our proposal is that lexical and dictionary stores can be brought together in an integrated federation of databases (also used in its broadest sense), so that we can query them by using a unique language, regardless of the particular resource to which the query will be finally targeted. As desirable as it might be, it is unrealistic to expect that the great variety of available lexical information resources could be converted into a single and standard representation schema in the near future. Hence, our approach aims for a solution that does not require any conversion of the sources, and that participates of the emerging "wrapper" technologies that enhance external communication with legacy systems (Roth and Schwartz 1997).

We address the problem of querying very different sources of lexical information —using for this purpose a unique and common query language— from the point of view of information integration. The so-called *local-as-view* paradigm is used for describing each lexical source as a view over a general conceptual model (GCM). So, it is necessary to specify a GCM to describe lexical knowledge, a source conceptual model (SCM) for each lexical source —expressed much in the way of the GCM—, as well as the manner to describe the content of each source in terms of the classes and relationships of the GCM —source content descriptions—. The GCM and some of the SCMs have been specified and implemented using a description logic language —*NeoClassic*—, and an answering algorithm that translates queries from the general model into each particular schema has also been implemented.

After this introduction, an outline of the architecture proposed is given in Section 1. In Section 2, the GCM for describing lexical knowledge is presented. Next, in Section 3, we discuss how to represent the particular conceptual models of the lexical sources (SCMs), and how to specify the sources in terms of the classes and relationships of the GCM, following the *local-as-view* approach. Next, in Section 4, we present the query translation algorithm used to convert queries made to the general model into particular queries addressed to the sources. Finally, an example of query is given in Section 5, and some conclusions are drawn.

1 An architecture for a federation of heterogeneous lexical databases

The most important features of the architecture proposed in this paper are (1) the incorporation of all the different lexical information sources into a federation of databases (the term database is here used in its widest sense), and (2) the conception of a general conceptual description model that allows users and/or programs to query the system regardless of the particular source to which the query will be finally addressed.

The general conceptual model provides the user with a global view of the information the system contains, stating it by means of a taxonomy of interrelated classes, in an object-oriented fashion. This global model has to be provided in terms of representation mechanisms that allow us to abstract from the physical and logical structure of data in the sources.

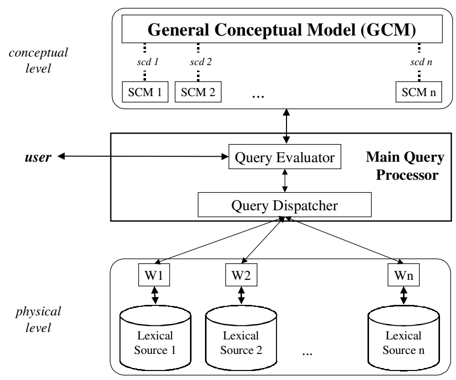

Figure 1. General view of the architecture of the system.

The sources are autonomous, and highly heterogeneous in the sense that they use very different data models and vocabularies. You could find among them from relational databases up to object-oriented knowledge bases, ontologies such as WordNet, or SGML-coded text documents like TEI-conformant MRD's. They are also heterogeneous from the point of view of the data they contain, because, as we have pointed out in the introduction, we are considering here lexical databases, electronic dictionaries, lexica used in NLP tasks, or even language-processing programs such as lemmatizers or part-of-speech taggers. The important thing here is to note that all the information resources considered contain closely related data: lexical data.

The *local-as-view* approach used in the field of data integration requires each source in the integrated system to be defined as a view of the global model, that is, in terms of the general conceptual description of the domain (see Section 3).

In Figure 1, a general view of the architecture of the system is given. At the top, the GCM, the SCM of each source, and the content descriptions of these sources (scd_i, represented in the figure by means of dotted lines) are depicted. The physical level, that is, the actual lexical resources, are illustrated at the bottom of the diagram. In the middle part we can see the information flow (note the arrows) between the user, the main query processor, and the wrappers of each source (w_i).

2 A general conceptual model for describing lexical knowledge

If you want a conceptual model for describing lexical knowledge to be general, this must be necessarily simple. But, at the same time, it must gather all the features and relationships that a lexical information source can establish.

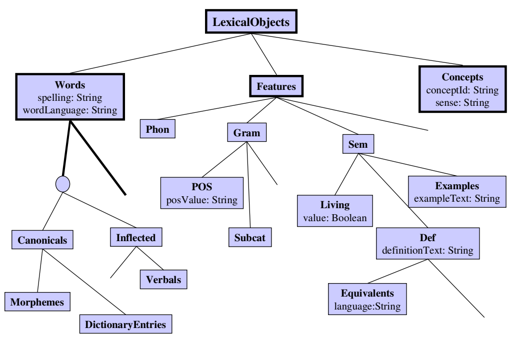

Figure 2. Partial view of the general conceptual model for describing lexical knowledge.

In our model, lexical knowledge is described by means of a taxonomy of classes and by a set of binary relationships established among the different objects or classes in the taxonomy. With regard to the taxonomy, you can find at the top a main class called *Lexical Objects,* and three subclasses of it: *Words, Concepts* and *Features* (see Figure 2). As for the relationships, a wide set of relations that link different classes in the hierarchy has been established and defined by specifying the domain, the range and the maximum cardinality of each relationship (see Figure 3). We can only refer here to a maximum cardinality of each relationship because, in the GCM, the relationships (and so the classes) are *virtual* in the sense that they don't actually store any information at all (the GCM is an abstract model).

Relation	Domain	Range	Max. Card.
senses	Words	Concepts	n
synonyms	Concepts	Concepts	n
pos	Concepts	POS	1
definitions	Concepts	Definitions	n
lemma	Inflected forms	Dictionary Entries	n

Figure 3. Some relationships established among the classes in the general conceptual model.

3 Describing different lexical sources according to the local-as-view paradigm

The different resources integrated in the system are pre-existing lexical information sources, and we want the system to allow to take in new sources dynamically. Moreover, user-posed queries are to be reformulated and converted into relations of the source databases in such a way that we can guarantee that the answers satisfy the constraints the queries originally had. Therefore, we will need rich mechanisms for describing the different conceptual models of the underlying data sources.

We describe two types of information for each source: the conceptual model of the sopurce (3.1) and the description of the contents (3.2).

3.1 Source Conceptual Models (SCMs)

At this level, the conceptual model of every resource is described. These SCMs will be highly heterogeneous, just like the data are.

The underlying structure and physical organization of the information sources are expected to be very diverse but, due the "wrapper" technology we have adopted, we can make an abstraction, and conceive and model an information source as a collection of objects belonging to some source relations. So, like in the GCM, every source model will be described by means of a taxonomy of classes and by a set of binary relationships among them. Note that now we can also describe the minimum cardinality for any source relation because they represent *real* relations, as opposite to the GCM where the relationships (and classes) are *virtual*, e.g. they don't actually store any information at all.

```
(createConcept LexicalObjects ClassicThing)
(createDisjoinGroup LexicalObjects oleDjGroup)
(createRole spelling true)
(createRole wordLanguage true)
(createRole senses)
(createConcept Words
   (and LexicalObjects
        (all spelling String)
        (all wordLanguage String)
        (all senses Concepts)
        (...))
   oleDjGroup)
(...)
(createRole conceptId true)
(createRole pos)
(createRole language)
(createConcept Concepts
   (and LexicalObjects
        (all conceptId String)
        (all pos POS)
        (...))
   oleDjGroup)
```

Figure 4. General Concept Model description in *Neoclassic*.

In principle, we could use arbitrary formulas in first-order logic to describe the data sources, but in such a case, a sound and complete reformulation would be practically impossible (see Section 4). The use of Description Logics to express domain knowledge is widely explored in the literature (Calvanese *et al.,* 1999); (Levy *et al.,* 1996). It is the formalism we have chosen to describe the different conceptual models, which have been implemented using the *Neoclassic* Description Logic Language. This language is used to model domain knowledge and it belongs to the family of KL-ONE knowledge representation languages. Thus, *Neoclassic* gives

us most of the representation power available in semantic networks. Although its expressive power is somewhat limited, it guarantees that certain basic operations (like testing if certain individual subsumes another one) are satisfiable in a reasonable time (McGuinness and Bodiga 1995).
Figure 4 shows the *Neoclassic* code that describes some of the classes and relations of the general conceptual model illustrated in Figure 2.

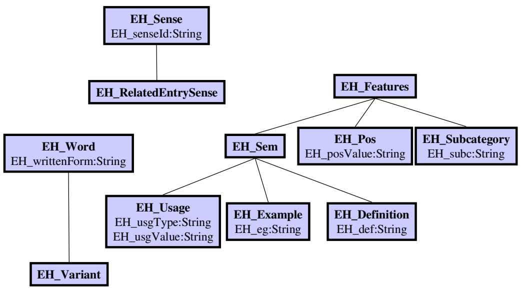

Figure 5. Partial view of the EH dictionary conceptual model.

In Figure 5 we see a partial view of the conceptual model of one of the sources. It represents some of the classes and relations occurring in the Euskal Hiztegia (*EH*) Basque monolingual dictionary, coded in SGML according to TEI.

3.2 Content description of the sources

At this level we describe the mappings between each SCM and the GCM. We adopt the *local-as-view* paradigm where each source relation (and class) is mapped into the general conceptual model.
For every relation (and class) in the local schema, a conjunctive query over the GCM must be specified. This query describes the conditions the tuples in the relation must satisfy[1]. This approach gives us two main advantages:

- The GCM does not change when a new resource is integrated into the system. This property allows us to offer the user a unique domain and vocabulary to pose the queries. It also leaves the way open to dynamically add new databases to the system.
- We can make fine-grained distinctions between the several relations and concepts of the sources, giving to the system the possibility of integrating very diverse lexical resources.

A conjunctive query Q is a conjunction of the form:

$$Q(\overline{Y}) \leftarrow \exists \overline{X} p_1(\overline{X_1}) \wedge ... \wedge p_n(\overline{X_n})$$

where Q is a relation (or class) in the source conceptual model, $\overline{X}, \overline{X_1}, ..., \overline{X_n}$ are tuples of variables or constants, and the p_i's are any of the relations in the GCM. The variables in Y, called the *distinguished variables*, are the variables that appear in $\overline{X_1} \cup ... \cup \overline{X_n}$ but not in $\overline{X}$, and thus they are the only variables in the query that are not existentially quantified.
In Figure 6 we can see some conjunctive queries that map some classes and relations of the EH dictionary resource to the GCM.
Classes are expressed as logical predicates with arity 1, and relationships are described by predicates with arity 2 (binary relations). So, the first query of the example can be read as follows: an individual `a` will be an `EH_Sense` instance in the EH source conceptual model if it is a `Concepts` instance in the GCM. The third query states that two individuals `s` and `l` will be related via the `EH_writtenForm` relation in the EH source conceptual model if `s` is a `Concepts` individual, `l` is a `DictionaryEntries`, and both instances are related via the `sense` relation in the GCM.

```
EH_Sense(a) <- Concepts(a).
EH_senseId(a,s) <- Concepts(a), conceptId(a,s).
EH_writtenForm(s,l) <-
  DictionaryEntries(l), sense(l,s), Concepts(s).
```

Figure 6. Mappings between the SCM of the EH dictionary and the GCM.

Note that the source doesn't need to contain all the objects that satisfy the query as the query only gives the necessary (but not sufficient) condition for the individuals. This led us to adopt the so-called *open-domain assumption*, i.e. we left the way open for other objects besides those described by the sources to exist. In fact, in the field of lexical information, the existence

[1] In contrast to the *global-as-view* paradigm where, for each general relation, we would write a query over the source relations specifying how to obtain general objects from the sources.

of a source that contains information about a certain concept doesn't obviously guarantee that all the lexical information about that concept is present in it. There can always be alternative sources that provide us with additional information on the concept.

4 The query translation algorithm

The query-answering algorithm guarantees that the descriptions of the sources are exploited in such a way that only sources that are relevant to a given query are effectively accessed.

If we want the query language of the system to be common, the user queries must be posed in terms of the GCM relations. Then the system must translate these queries into the equivalent rewritings by means of the SCM of each lexical source. The queries must be rewritten in such a way that we can obtain as many answers as possible from a set of data source descriptions.

This problem is closely related to the "answering queries using views" problem (Levy *et al.,* 1995) that can be stated as follows: given a query Q formulated in terms of the database schema, and a set of view definitions $V_1,...,V_n$ over the same schema, answer the query Q using only the answers to the views $V_1,...,V_n$. In our case, the view definitions are just the content descriptions of every relation (class) in the local sources.

Two aspects are to be taken into account when dealing with this problem: the so-called *sound* and *complete* reformulations. Since we are constructing query rewritings for some set of views, we must verify that this rewritings will effectively give us the required information (i.e. the same objects that the original query would). Informally speaking, a reformulation is said to be *sound* if all the answers it produces are guaranteed to be answers to the original query. On the other hand, a rewriting is said to be *complete* if when an object is not in the answer we can conclude that it does not exist. A rewriting is said to be *exact* if it is *sound* and *complete*.

Clearly, we are interested in finding a set of *sound* conjunctive rewritings for some given query, since we adopted the *open-domain* model (see 3.2).

In Levy *et al.* (1996) and Duschka and Genesereth (1997) we can find some algorithms to answer queries using views in the context of data integration. We have borrowed the so-called *bucket algorithm* originally developed as part of the Information Manifold system (Levy *et al.,* 1996). The bucket algorithm finds the relevant sources for a given query Q and reformulates them exclusively in terms of those source models. It also guarantees that this rewritings are the *maximally-contained rewritings,* i.e. that they provide all the possible answers for a given set of sources.

The algorithm proceeds in two steps. In the first step, it computes for each subgoal in the query the sources that are relevant to it. The knowledge stored in the logical description is extensibly used for accomplishing this task. In a second step, it considers rewritings constructed by choosing one relevant source for every subgoal in the query, and checks if they are contained in the original query. To perform this task, it expands every subgoal in the rewriting with the content description associated to it, renaming the variable names as required. The expanded rewriting refers only to GCM relations (classes) and the algorithm tests its satisfiability with the model and wether it is contained in the original query. This can be performed and implemented using *Neoclassic*.

The problem of answering queries using views can be undecidable if conjunctive queries with negations or recursion are used to define the view definition rules (Abiteboul *et al.,* 1998). However, we use neither negated nor recursive predicates in our system and, according with Levy *et al.* (1995), we only need to consider rewritings the length of which is at most the length of the original query. Therefore, we can assure that the algorithm will produce semantically equivalent rewritings.

5 Querying the system: an example

As an example, we will pose a query asking for all the possible categories for a given word "lagun"[2].

In the GCM, this query can be expressed as:

```
q("lagun", x) <- DictionaryEntries(h),
    spelling(h, "lagun"), senses(h,a),
    pos(a,lc),posValue(lc, x).
```

which states that `h` is a `DictionaryEntries` object, `spelling` attribute of which is the required word; then, it ties the `h` entry with all

[2] Basque word for " friend"

its possible concepts via the `senses` relation, and it retrieves for each one the grammatical information (via the `pos` relation). Finally it asks for its POS values (`posValue` attribute).

In a first step, the algorithm uses the content description of each source and builds one bucket per source with all the relevant relations for every subgoal of the query. Then it combines these source relations and returns all the possible rewritings for each source. Figure 7 shows the rewritings for the EH lexical source.

As explained before, the algorithm expands every rewriting by substituting each subgoal with its associated content description. Thus, each rewriting can be considered a representation of some related instances in the GCM. So, in a final step, the algorithm determines for each rewriting if its expansion is allowed in the GCM. The expansion of rewriting 1 will be discarded because if the predicate `EH_pos(a,lc)` is true then (conforming its content description conjunctive query) `lc` must be a `POS` object. Similarly, the expansion of the next predicate, `EH_usgValue(lc,x)`, states that `lc` is an `Usage` object. But, since `POS` and `Usage` are two disjoint classes in the GCM, this representation is invalid in the model.

```
Rewriting: 1 -- FAIL --
Q'("lagun",x) :- EH_spelling(h,"lagun"),
EH_variants(a,h), EH_pos(a,lc), EH_usgValue(lc,x)


Rewriting: 2 -- FAIL --
Q'("lagun",x) :- EH_spelling(h,"lagun"),
EH_variants(a,h), EH_pos(a,lc), EH_posValue(lc,x)


Rewriting: 3 -- GOOD --
Q'("lagun",x) :- EH_spelling(h,"lagun"),
EH_sense(h,a), EH_pos(a,lc), EH_posValue(lc,x)


Rewriting: 4 -- FAIL --
Q'("lagun",x) :- EH_spelling(h,"lagun"),
EH_sense(h,a), EH_pos(a,lc), EH_usgValue(lc,x)

Rewriting: 5 -- FAIL --
Q'("lagun",x) :- EH_sense(h,a),
EH_spelling(h,"lagun"), EH_pos(a,lc),
EH_usg_value(lc,x)
...
```

Figure 7. Original rewritings for the EH source returned by the bucket algorithm. Those marked with – FAIL – are invalid rewritings.

Rewriting 3 is the only representation allowed in the GCM and, thus, it is also the only rewriting the algorithm will finally return for this source. It is also the query that we would pose to the SCM of EH if we wanted all the possible categories for one given word to be retrieved.

Conclusion

A solution for the problem of integration of diverse lexical resources has been presented. This solution aims to use data integration techniques in the field of lexical knowledge.

The architecture proposed allows the user —be it human or a computer program— to pose queries to the system regardless of the actual formats and representations used in the particular resources integrated in it. This is possible due to the existence of a general conceptual model of lexical knowledge. The architecture allows the integration of new resources to the system in a dynamic way. A Conceptual Model for describing general lexical knowledge has also been presented.

Currently two lexical resources have been integrated in our system. One is a SGMLized TEI-conformant Basque monolingual dictionary called *Euskal Hiztegia* (EH). The EH dictionary (Sarasola, 1995) is a general purpose monolingual dictionary, and it covers standard Basque. The structure of an EH entry is: headword; date; variants; part of speech; abbreviations (style and usage labels, field labels, etc.); definition; relations; scientific names; examples; subentries and grammatical information. It contains 33,180 entries and 41,699 senses. It is locally stored as a collection of SGML documents. The second one is a *pure* lexical database called EDBL (Aldezabal *et al.*, 2001), used in language-processing tasks and stored in a classic relational DBMS. EDBL is a large source of lexical information (with about 80,000 entries) and has been designed as a multipurpose lexical support for the different processes involved in the automatic treatment of Basque.

We are currently designing a graphical user interface that will permit the human user to pose queries to the system in a friendly way.

References

Aldezabal I., Ansa O., Arrieta B., Artola X., Ezeiza N., Hernández G., Lersundi M. (2001) *EDBL: a General Lexical Basis for the Automatic Processing of Basque.* IRCS Workshop on Linguistic Databases. December 11-13. Philadelphia.

Abiteboul S., Duschka O. (1998) *Complexity of Answering Queries Using Materialized Views*. In "Proceedings of the 17th ACM SIGART-SIGMOD-SIGART Conference on Principles of Database Systems (PODS)", pp. 254-263. Seattle, WA.

Calvanese D., de Giacomo G., Lenzerini M., Nardi D., Rosati R. (forthcoming) *Data Integration in Data Warehousing*. Int. Journal of Cooperative Information Systems. World Scientific Information Systems.

Calvanese D., de Giacomo G., Lenzerini M. (1999) *Answering queries using views in description logics.* In, "Proceedings of the 6th International Workshop KRDB'96", pp. 6-10. E. Franconi and M. Kifer, eds.

Duschka O., Genesereth M. (1997) *Answering recursive queries using views*. In "Proceedings of the ACM Symposium on Principles of Database Systems (PODS)", pp. 109-116.

Levy A.Y., Mendelzon A.O., Sagiv Y. (1995) *Answering Queries using Views*. In "Proceedings of the ACM Symposium on Principles of Database Systems (PODS)", San Jose.

Levy A.Y., Rajaraman A., Ordille J.J. (1996) *Querying Heterogeneous Information Sources Using Source Descriptions*. In "Proceedings of the 22nd International Conference on Very Large Databases (VLDB)", pp. 251-262, Bombay, India.

McGuinness D.L., Borgida A. (1995) *Explaining subsumption in Description Logic.* In "Proceedings of the 14th International Joint Conference on Artificial Intelligence, (IJCAI'95)", pp. 816-821, Montreal, Canada. Morgan Kaufmann.

Patrick J., Zhang J., Artola X. (2000) *An Architecture and Query Language for a Federation of Heterogeneous Dictionary Databases*. Computers and the Humanities 34 (4), pp. 393-407.

Sarasola I. (1995) *Hauta-Lanerako Euskal Hiztegia*. Gipuzkoako Kutxa. Donostia.

Roth M., Schwartz P.M. (1997) *Don't Scrap it, Wrap it!, A Wrapper Architecture for Legacy Data Stores*. In "Proceedings of 23rd International Conference on Very Large Databases (VLDB'97)", pp. 266-275. August 25-29, Athens (Greece).