From Natural Language to Databases via Ontologies

Leonardo Lesmo, Livio Robaldo

Dipartimento di Informatica - Università di Torino

Corso Svizzera 185 – 10149 Torino – Italy.

E-mail: lesmo@di.unito.it, robaldo@di.unito.it

Abstract

This paper describes an approach to Natural Language access to databases based on ontologies. Their role is to make the central part of the translation process independent both of the specific language and of the particular database schema. The input sentence is parsed and the parse tree is semantically annotated via references to the ontology describing the application. This first step is, of course, language dependent: the parsing process depends on the syntax of the language and the annotation depends on the meaning of words, expressed as links between words and concepts in the ontology. Then, the annotated tree is used to produce an "ontological query", i.e. a query expressed in terms of paths on the ontology. This second step is entirely language- and DB-independent. Finally, the ontological query is translated into a standard SQL query, on the basis of a concept-to-DB mapping, specifying how each concept and relation is mapped onto the database.

1. Introduction

One of the most relevant problems in data access is today the one of model management. It concerns the correspondence between *models*, where a model is a specific way to describe a set of data and the way the data are represented inside a computer. Consequently, *model* is a term that encompasses DB schemas, XML schemas, and ontologies. The mappings between models can be viewed from two different perspectives. The first of them concerns the ease with which the correspondences between models is maintained. It is clear that a given model may evolve through time, so that a change in a model implies a corresponding change in the mapping to any related model (Melnik et al. 2003).

The second perspective from which the mapping between models is studied concerns its use. Such a mapping, in fact, has the goal of enabling a "translation" between expressions (e.g. queries) related to one model into equivalent queries related to the other model. One of the goals of this translation is the possibility of "populating" an ontology with instances extracted from a database. In order to do this, one must know which concepts in the ontology correspond to which tables, rows, and columns in the database (Bizer 2003), (D2R 2005). In some cases, automatic approaches for establishing the mapping have been proposed (Stojanovic et al 02); however, they rely on a strong similarity between the schema of the database and the organization of the ontology, so that manual approaches have been proposed to describe the mapping (Barrasa et al. 2004).

This paper addresses the problem of defining and using the mapping as a way to access in natural language the contents of a database. We take an ontology as the basic repository of domain knowledge. The underlying ontology is all the system knows about the domain, so it is the basic knowledge source enabling the system to understand the goals of the user. The system "thinks" in terms of ontology concepts and relations and it expresses the goals of the user in terms of expressions we call "ontological queries". It is assumed that the ontology is language-independent, so that the user may speak English, or Italian, but the resulting ontological queries are equivalent. Note that it is not necessary that the ontology of the user (i.e. her/his knowledge of the domain) corresponds exactly to the system's ontology: it is sufficient that the correspondence is strict enough to enable the system to respond appropriately to the user's inputs. The needed degree of correspondence is an empirical matter that could be investigated on the basis of the pragmatic appropriateness of the system's behaviour.

So, the system knows the language well enough to interpret the input sentences in terms of concepts in the ontology, but it must also know about the organization of the database: it acts as a human expert that supports the user in writing the correct DB query for getting the data. This knowledge is expressed as a mapping between models, i.e. the ontology and the DB schema.

This paper aims at describing the way the language is interpreted in terms of an underlying ontology, and the way this interpretation is used to build the correct DB query. In fig.1, we report the architecture of the NaLDaB system. Actually, the real implementation is simpler, since

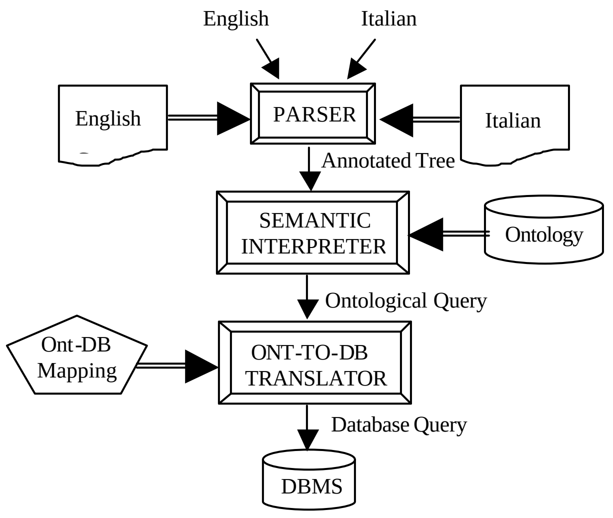

Figure 1: The architecture of the system

we currently have just one repository of syntactic knowledge (common to Italian and English) enabling the system to build the parse tree for both languages. On the other hand, this KS can hardly be called a grammar, since it is just a set of condition-action rules specifying the most probable attachment of words in a Dependency Grammar framework. The parser is not the focus of this paper, so that the parse tree is taken here as the input of the process of DB access.

2. Input

The input to the whole process is a parse tree representing the structure of the input sentence (some details about the parser are reported in (Lesmo et al. 2002) (Bosco & Lombardo 2003)). The parse tree is a dependency tree, i.e. a tree such that, approximately, each word of the input sentence corresponds to a node of the tree, and viceversa. Dependency parsing has received great attention in the linguistic literature (e.g. (Mel'cuk 1988) (Hudson 1990)); among its features, the one which is most relevant here is the strict correspondence between a dependency tree and a predicate-argument structure.

In this paper, we will use the following example.

- *Puoi dirmi dove è la biglietteria di Settembre Musica?*[1] (Can [you] tell-me where is the ticket-counter of Settembre Musica?)

The dependency tree is shown in fig.2. In each node, we reported the input form (word), the lemma, and the corresponding English translation. In the tree, there appear two traces, which are inserted to refer to the understood subject of the sentence (pro-drop in Italian) and to the shared subject between a verb (to tell) and its governing modal (can). They are automatically inserted in the tree by the parser. The labels on the arcs are organized in order to mark the syntactic/semantic role of the dependent with respect to the governor.

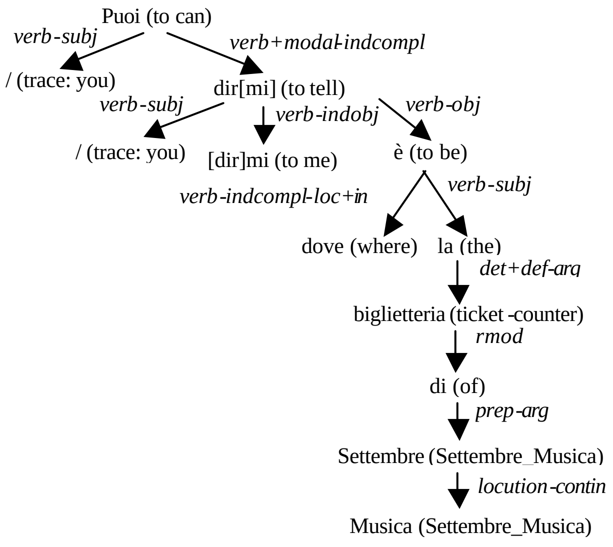

Figure 2: The dependency tree of "Puoi dirmi dove è la biglietteria di Settembre Musica?"

[1] "Settembre Musica" is the collective name of a group of concerts held in September in Torino. It is a complex event, in the sense that it has its own organization (and a ticket counter).

3. Knowledge bases

3.1 The Ontology

A very simple ontology has been implemented in OWL (McGuinnes et al. 2004). In OWL, both subclass relations and properties of the individuals belonging to a class (i.e., OWL restrictions) are asserted by means of the mathematical relation $\subseteq$. Nevertheless, we adopt here a graphical representation similar to KL-one (see fig.3) in order to increase readability[2].

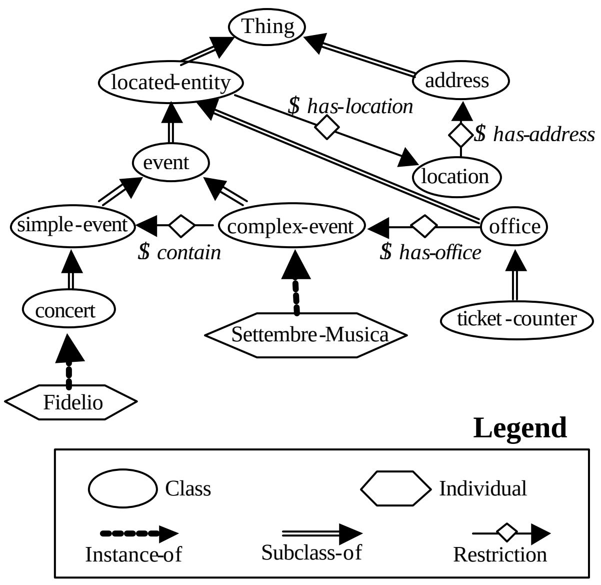

Figure 3: Graphical representation of the (partial) ontology

3.2 The meaning of the words

This is simply expressed as a mapping between words and concepts (nodes) in the ontology. It is just a set of pairs *<word concept>*, as shown in Tab.1,.where the ££ prefix marks concepts in the ontology.[3]

Italian	English	Concept
potere	can	££can
dire	tell	££tell
biglietteria	ticket-counter	££ticket-counter
dove	where	££location
…	…	…

Table 1: The mapping from words to concepts.

[2] The ontology includes just some individuals. for example, the individuals corresponding to the concerts (e.g. *Fidelio*), the complex events (e.g. *SettembreMusica*), etc., but not the ticket counters or the addresses: we assume that a user could ask, for example, "Can [you] tell-me where is the ticket-counter of Settembre Musica?", but not "Which complex event does correspond to the ticket counter in via Verdi 12?". Moreover, no information is present concerning the relations among them: these data are assumed to be present only in the DataBase.

[3] Very little has been done to date to handle non-content words (as articles), which deserve further study. The correspondence between the English dictionary and the concept names is only aimed at enhancing readability: nothing changes if a name such as *££location* is consistently changed into *&%$X7723*

3.3 The Ontology-to-Database mapping

This mapping (OD mapping) enables one to state how a given concept or a given relation is represented in the backend database. This is obtained by associating with every element (concepts and relations) of the ontology a description of the mapping (see fig.4).

- The *T* prefix identifies tables, while *A* refers to attributes (columns).
- "distr" means that a concept or relation is split over more than one table.
- For relations, "eq" means join over two tables.
- "path" specifies that the DB definition covers the whole path from the referred concept (or relation) to the end of the path. So, in the example, to cover the ontological path leading from *&has-location* to *&address-description* it is enough to use either the **A*org-addr* attribute (for organization) or the **A*place* attribute (for events).

The three examples of relations reported in fig. 4 cover increasing levels of complexity. The first one (*&has-address*) is a simple identity function. In fact, in the ontology, the concepts associated with the location (*££location*, which is the range of *&has-address*) and with its address conflate into the same datum of the DB (a location "is", in our simplified database, the string that describes its address).

```
(££event *T*main-agenda)
(££office *T*organiz (eq *A*org-class office))
(££ticket-counter *T*organiz (eq *A*org-serv tick-count))
(&has-address
      (distr  (rdef     (range *T*organiz *A*org-addr)
                        (domain *T*organiz *A*org-addr))
              (rdef     (range *T*agenda *A*place)
                         (domain *T*agenda *A*place))))
(&has-location
      (path forward (££locatio n &has-address ££address))
      (distr  (rdef     (range *T*organiz *A*org-id)
                        (domain *T*organiz *A*org-addr))
              (rdef     (range *T*agenda *A*event-id)
                        (domain *T*agenda *A*place))))
(&has-office
      (distr  (rdef    (range *T*organiz *A*org-id)
                       (eq     (*T*organiz *A*org-id)
                               (*T*organiz *A*off-of) )
                       (domain *T*organiz *A*org-id))
              (rdef    (range *T*main -agenda *A*event-id)
                       (eq     (*T*main -agenda  *A*event-id)
(*T*organiz *A*off-of))
                       (domain *T*organiz *A*org-id))))
```

Figure 4: The mapping between concepts in the ontology and Database structures

The second relation (*&has-location*) is solved in a single-relation access, but not as an identity. In fact, the "location" of an organization is retrieved from the DB by inspecting the *organiz* table, by using as access key the *org-id* attribute, and by extracting the *org-addr* attribute. Finally, the *&has-office* relation involves a join over two tables. This is expressed by stating the "external" attributes referring to the range and domain of the relation (*range* and *domain*) and the join attributes (*eq* …). Of course, this should be extended to cover more complex inter-table relationships.

3.4 The Database

The part of the data base schema relevant for our example is reported, with some comments, in fig.5. Note that in the schema, the *T* and *A* prefixes do not appear.
It includes three tables (*agenda*, *main-agenda*, and *organiz*), which describe, respectively, single events (e.g. a concert), set of events (e.g Settembre Musica), and organizations. The latter may be cultural organizations or other kinds of organizations (as the Public Transport Agency, or any kind of offices). In order to make the things more intricate, we assumed an internal join in the *organiz* table, enabling one to extract the offices of a given (cultural or other) organization. Clearly, this is a bad DB design, but everybody knows that real DB's are often much worse than this.

4. The interpretation process

The construction of the query is based on a separation between the topic and the focus of the sentence. For instance, in our example, "Dove" is the topic, i.e. the element asked about, while the focus is "La biglietteria di Settembre Musica". Usually, the topic is marked by a question element, i.e. an adverb (as in "Dove", or a question adjective, as "Quali concerti", 'which concerts"), while the focus is given by all other dependents (except punctuation marks and auxiliaries). However, in order to build up the ontological query, it is more useful to think in terms of "goal" and "restrictions". The goal consists in the semantic "head" of the focus, while the restrictions are determined on the basis of possible modifiers of the head plus other components of the sentence. For instance, in

- *Quali concerti diretti da Abbado ci sono al Regio? [Which concerts conducted by Abbado are there at the Regio?]*

we have that the goal is "Concerti" (or, more properly, the concert identifiers), while the restrictions include both "diretti da Abbado" (conducted by Abbado), which is a

```
(db-name cultural-agenda-v0
     (table agenda
        *** these are specific events (a concert, an exposition, ...)
                (event-id integer p-key)
                (event-name string)
                (event-type string)            *** concert, movie, ...
                (place string)                 ***an address
     ….. )
     (table main -agenda
                (event-id integer p-key)
                (event-name string)
                (event-type string))
     (table organiz
                (org-id integer p-key)
                (org-name string)
                (org-class string)      *** instit, office, ...
                (org-type string)       ***artistic, commercial, ...
                (org-addr string)
                (org-serv string)       ***ticket-counter, information,
                (off-of integer s-key (organiz main -agenda))
     )
 )
```

Figure 5: The database Schema.

syntactic dependent of “concerti” and “al Regio” (at the Regio Theater), which is a dependent of the main verb. Finally, it is worth noting that some yes/no questions are (in some limited cases) handled as implicit requests of information. So, in

- *Ci sono concerti al Regio domani?*
  [Are there concerts at the Regio tomorrow?]

In case the answer is affirmative, the list of relevant concerts are returned.

4.1. Annotation of the syntactic tree

This first subprocess, taking in input the syntactic tree, adds a pointer to an ontology concept for each lemma for which the mapping described in §4.2 is defined. Moreover, it reorganizes, when necessary, the tree by solving co-references. Concerning sentence 1), this subprocess adds the semantic pointers to the lemmas *where (££location)*, *biglietteria (££ticket-counter)* and *Settembre (£Settembre-Musica)*[4] and associates the two traces (*you*) and the clitic (*to me*) to the deictic elements *§myself* and *§speaker*, respectively.

4.2 Top level elements.

The input sentence often includes elements which are not useful for determining topic and focus, as, for instance, "Puoi dirmi" (can you tell me), "Vorrei sapere" (I would like to know), "per favore" (please). Some of these elements are governors of the actual query sentence (i.e. they occur higher in the parse tree). Others (as 'please') depend on the main verb of the sentence. The first step of the translator is to travel down the tree in order to skip the upper elements. When the actual top verb (the useful one) is found, then the query elements are sought below it. So, the overall organization can be depicted as in fig.6. The topic and the focus are interpreted separately, but some join element is used. In this example, the join element is

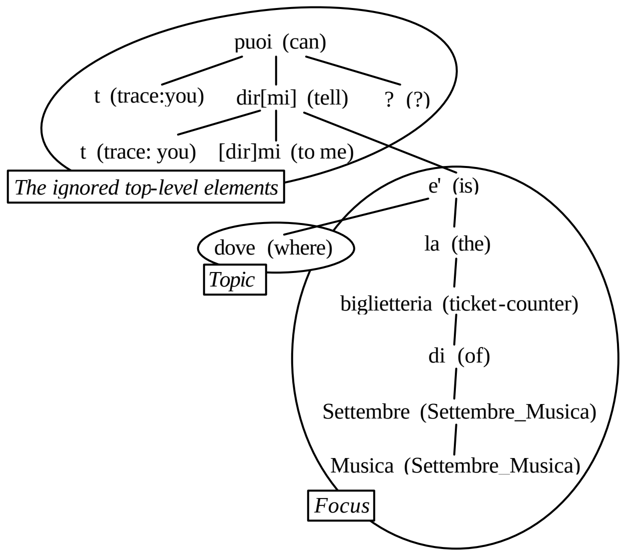

Figure 6: Actual fragment of parse tree used in semantic interpretation

[4] No interpretation is associated to *Musica*, since it is taken just as the continuation of a Multiword: all relevant information is associated with the first word of the sequence (*Settembre*).

the word “biglietteria”, which, in the tree, has been associated with the concept *££ticket-counter*, during the annotation phase. Intuitively, the sentence is interpreted as follows:

a) what is desired is the “where” of some “ticket-counter”
b) the “ticket-counter” is the one of “Settembre Musica”

Now, the problem is to find two subpaths, the first describing how to reach information useful for the user from the “where of ticket-counter” description, the second linking ticket-counter to Settembre_Musica. We adopted two different solutions for the two tasks.

As concerns the path from a concept to a “goal” (i.e. useful information) we adopted a knowledge-intensive solution, in the sense that an additional knowledge repository must be built that contains information about the reasonable answer. In the case of the example, it specifies the path *(££location) &has-address (££address)*. This is intended to mean that when an inquiry refers to a location, it is an implicit request of its address.

For what concerns the “restriction” of the query, i.e. the part regarding the fact that the location is the one of a ticket counter (and that this latter is the one of Settembre Musica), we adopted a substantially different approach, i.e. we assumed that it is possible to automatically find a path from the concept *££location* and the concept *££ticket-counter* (and from *££ticket-counter* to *£Settembre_Musica*). This is carried out by looking for the shortest path connecting the two concepts in the ontology (without counting *subclass-of* links, in order to account for inheritance)[5]. In our example ontology, this path is (fig.3).

```
(££location)
    &is-location-of (££located-entity)
        has-subclass (££office)
            has-subclass (££ticket-counter)
```

The final ontological query obtained for the example is reported in fig.7. Note that the default query target has been split into two parts: The first one (consisting in the relation leading to *$string*) has been left as the “select” clause, while the second one has been inverted and attached as the first part of the restriction: what the user actually is assumed to desire is an address, while all the rest says which address s/he wants.

```
SELECT   (££address) &is-address-of
FROM     (££location) ticket-counter
WHERE    (££location) & is-location-of (££located-entity)
    has-subclass (££office) has-subclass (££ticket-counter)
    is-subclass-of(££office) is-office-of (££complex-event)
    has-instance (£Settembre_Musica)
```

Figure 7: The ontological query for the example

4.3 Construction of the Database query

This process is based on the Ontology-to-Database mapping described in §4.2. The basic idea is to follow the

[5] A problem that deserves further investigation concerns the presence, in the input sentence, of prepositions. We currently adopted “preplate tables”, i.e. contextualized references to concepts in the ontology. Because of space constraints, we cannot describe them here.

path on the ontology and, for each step, build a corresponding piece of database query. The "select" part and the "where" part of the ontological query are handled separately, and they are composed at the end of the process.

4.3.1 Moving one step

A step on the ontological path is defined as a triple:

<concept1 relation concept2>

For instance, in fig.8, we see the path *<££office &is-office-of ££complex-event>*; note that this path, because of inheritance, subsumes the (valid) path *<£$ticket-counter &is-office-of ££complex-event>*
The following information is extracted from the Ontology-to-Database mapping (§3.3):

a. The definition of *concept1*, in the form (see §3.3):

   <table-name conc-restriction>

   where *conc-restriction* may be absent (ex. *<*T*main-agenda>*, for *££event*) or have the form *<op attr val>* (*<*T*organiz (eq *A*org-serv tick-count)>* for *££ticket-counter, see fig.8*)

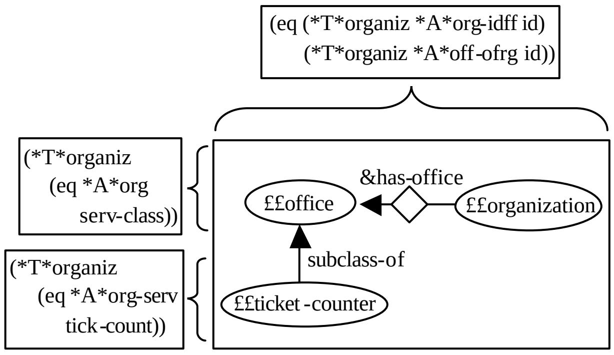

Figure 8: A path in the ontology and its mapping to the DB.

b. The definition of *relation*, given as

   <relation-name [path-def] actual-def>

   where the optional *path-def* is:

   <concept1 relation1 concept2 relation2 …>

   and *actual-def* is

   <basic-def | <distr basic-def1 basic-def2 …>>

   finally, a *basic-def* is:

   <eq <table1 attr1> <table2 attr2>>

   or

   <table attr>

Intuitively, a basic definition (*basic-def*) specifies how a relation is implemented in the database in case no ambiguity is present. This can happen via a join operation, so that the relation *&organized-by* (defined over an event and an organization) can specify that the data about the organization can be obtained by joining the tables *agenda* (where the organizer id appears) and *organiz* (where all the data about the organizing institution may be found). Otherwise, all the data may be available inside a single table. For instance, if no attributes are specified for a *££location*, which in the database may simply appear as a string (the address), then the location of an event may be obtained by inspecting the *agenda* table.

On the basis of these data, the relation definition most appropriate for *concept1* is extracted. Now, it is possible to compose the query, which will have the form:

<external-table-name external-attribute external-restriction internal-restriction>

- the external-table-name is the one associated with the concept and with the involved "direction" of the relation (**T*organiz* in fig.8).
- the external-attribute is the one associated with the involved "direction" of the relation (**A*off-of* in fig.8)
- the external-restriction is the one associated with the concept definition (again **T*organiz* in the particular example of fig.8).
- the internal-restriction is the one associated with the "other direction" of the relation (*(*T*organiz *A*org-id)* in fig.8).

4.3.2 Composing steps

This operation is based on a recursive application of the single-step translation process mentioned above. The result of a single step is a full query. Consequently, the result of a recursive call on the remaining part of the ontological path (i.e. the one starting from *concept2*, having already solved the single step *<concept1 relation concept2 ...>*) is the specification of *concept2*, as provided by *<concept2 ...>*. If we apply the translation process to this remaining part, we obtain a full query specifying how to extract from the backend the instances of *concept2* relevant for the user. At this point, it is possible to use this query as a further restriction for what, in the previous paragraph, has been called *internal-restriction*. In our example, the required composition is depicted in fig.9. After removing some redundancies, as well as the table/attribute prefix, we obtain:

```
(organiz  org-id (eq org-serv tick-count)
        (eq off-of (organiz event-id
                (eq event-name Settembre_Musica))))
```

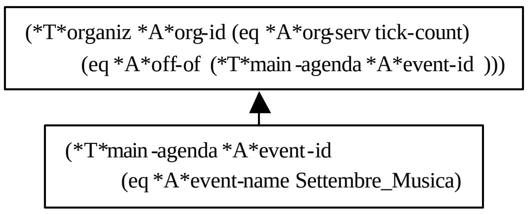
(*T*organiz *A*org-id (eq *A*org-serv tick-count)
(eq *A*off-of (*T*main-agenda *A*event-id)))
(*T*main-agenda *A*event-id
(eq *A*event-name Settembre_Musica)

Figure 9: Composition of queries.

4.3.3 Composition of *select* and *where* parts

In the previous paragraph, we have described in detail the processing of the *where* part of our example. The interpretation of the select part is exactly analogous and, in our example, it produces:

```
(organiz  org-addr (eq org-serv tick-count))
```

The merging of the two parts does not require particular comments, since it is easy to see that it may produce (after dropping redundant restrictions) the final query:

```
(organiz  org-addr (eq org-serv tick-count)
    (eq off-of (main-agenda event-id
      (eq event-name Settembre_Musica)) )
```

Which corresponds to the SQL query:

```
select org-addr
from organiz
where  org-serv =”tick-count” and
     off-of  in(select event-id
                from main-agenda
                where event-name= “Settembre_Musica”)
```

5. Ontology as Interlingua

In this paper, we have shown that an ontological query can be used as an abstract representation of the request of a user asking for information. It has been stated that the same ontological query can be obtained for Italian and English sentences, though it is clear that this only applies to rather simple sentences, used to query the content of a database.

In this application, it is assumed that the input is the natural language query, while the output is the SQL query. This is due to the fact that the focus of the paper is on the interpretation of language, and not on generation. However, the ontological query specifies the meaning of the NL sentence, in terms of the way the computer (on the basis of its ontology) is assumed to think about the domain of application. In other words, the "ontology" on which the input sentence is based is the humans' ontology, but the sentence is understood in terms of the computer ontology. In some relevant sense, the database schema is the DBMS ontology. So, the central step of the interpretation process may be seen as a translation between two points of view: the interpreter's point of view and the DBMS's point of view. The fact the what we call ontology has a more relevant role is only due to the (reasonable) assumption that the interpreter is the module which knows about the world (the domain of application).

A second comment concerns the different ways computers may "think" about the problem. In this paper, we focussed on the generation of DB queries, but this system has been applied within the HOPS project (see Acknowledgments), which is mainly devoted to dialogue management. This has two implications: the first of them is that the goals of the user may be expressed by means of more than one sentence (a sequence of steps in the dialogue); the second implication, perhaps more relevant here, is that in many dialogue systems the final query is not expressed in terms of a SQL-like language, but in a much simpler form, i.e. as a set of pairs parameter-value. This is especially true for speech interactions, and is what is enforced in the speech documents of W3C. So, we could say that some computers think in terms of tables and attributes, while others think in terms of parameters. The interesting point is that the same ontological query can be translated either in a database query or in a set of parameter-value pairs. This has actually been made in the current implementation: depending on the context (single sentence or dialogue), the system gets from the ontological query either the SQL query or the set of parameters. It could be said that (at least in the present implementation) the ontological query is an interlingua at work between two human languages (Italian and English) and two computer languages (SQL and parameters).

6. Conclusions

This paper describes a ontology-based system for translating NL queries into SQL queries. It has been shown that the translation process is independent both of the particular Natural Language and of the particular DB schema. Moreover, it has been noted that the ontological query that constitutes the intermediate representation that supports the translation process can be used also for producing different targets of the translation (parameters). The described system is a practical implementation of the basic role of ontologies, i.e. the one of enforcing interoperability: once the knowledge about the domain is represented in ontological terms the same process can be applied to any domain and to any language. Of course, the idea of an universal ontology has been challenged under various respects, first of all the practical impossibility of making people agree on the basic top-level concepts, although some theoretical works on the well-formedness of ontological top-levels can be viewed as a relevant step toward the construction of such a general ontology.

We agree on the practical impossibility of a single "standard" ontology, and the paper shows that such an ontology is not really necessary, but what is needed is a bridge between ontologies: natural language can help in defining such a bridge.

7. Acknowledgements

This work described has been carried out within the HOPS project funded by the EC (IST-2002-507967). HOPS means "Enabling an Intelligent Natural Language Based **H**ub for the Deployment of Advanced Semantically Enriched Multi-channel Mass-scale **O**nline **P**ublic **S**ervices" and is devoted to the access via speech and text to public services. Although the work described here has been entirely carried out at the Dip. di Informatica of the University of Torino, it could not have been possible without the continuous interaction with the partners in the project. All of them made significant contributions, but we want to mention here especially Oscar Corcho (ISOCO) and Marta Gatius (TALP, University of Barcelona). Also, Pietro Torasso (from our Department), Morena Danieli and Sheyla Militello (Loquendo S.p.A., Torino), helped us to clarify some points of the approach described herein.

8. References

J.Barrasa, Ó.Corcho, A.Gómez-Pérez (2004). R_2O, an Extensible and Semantically Based Database-to-Ontology Mapping Language.

C.Bizer (2003). D2R Map - A Database to RDF Mapping Language. Proc. WWW 2003, Budapest.

C.Bosco, V.Lombardo (2003). A relation-based schema for treebank annotation. In Cappelli, Turini (eds).: Advances in Artificial Intelligence, Springer Verlag, Berlin, 462-473.

D2R (2005). site: http://www.wiwiss.fu-berlin.de/suhl/bizer/d2map/D2Rmap.htm

R.Hudson, (1990). English word grammar. Basil Blackwell, Oxford and Cambridge, MA, 1990.

L.Lesmo, V.Lombardo and C.Bosco, (2002): Treebank Development: the TUT Approach, in Sangal and Bendre (eds.): Recent Advances in Natural Language Processing, Vikas Publ. House, New Delhi, 2002, 61-70.

D.L. McGuinness, K.Smith, C.Welty, (2004) Owl Web Ontology Language Guide, *http://www.w3.org/TR/owl-guide/*.

I.Mel'cuk. (1988): Dependency syntax: theory and practice, SUNY University Press.

S.Melnik, E.Rahm, P.A.Bernstein, (2003). Rondo: A Programming Platform for Generic Model Management. SIGMOD 2003, San Diego CA.