NLDB | 2003

June 23-25, Burg (Spreewald)
near to Cottbus, Germany

8th International Conference on Applications of Natural Language to Information Systems

home
topic
aims and scope
conference program
conference venue
travel information
photos of conference
conference organisation
contact

conference program

Sunday, June 22nd

19.00 Coming together

Monday, June 23rd

08.30 - 11.30 Applications of Information Retrieval
11.30 - 13.00 Advanced Modelling
14.30 - 16.00 NLDB Applications
16.30 - 18.00 Intelligent Querying

Tuesday, June 24th

08.30 - 10.00 Linguistic Resources
10.00 - 11.00 Information Extraction
11.30 - 13.00 Referencing and Categorization
15.00 - 18.00 Social Program

Wednesday, June 25th

08.30 - 09.30 Invited Talk
10.00 - 11.00 Linguistic Resources
11.30 - 13.00 Ontologies in the Web

NLIDB Templates for Semantic Parsing

Niculae Stratica, Leila Kosseim and Bipin C. Desai

Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, H3G 1M8, Canada
nstradi@sprint.ca
kosseim@cs.concordia.ca
bcdesai@cs.concordia.ca

Abstract: In this paper, we present our work in building a template-based system for translating English sentences into SQL queries for a relational database system. The input sentences are syntactically parsed using the Link Parser, and semantically parsed through the use of domain-specific templates. The system is composed of a pre-processor and a run-time module. The pre-processor builds a conceptual knowledge base from the database schema using WordNet. The knowledge base is then used at run-time to semantically parse the input and create the SQL query. The system is meant to be domain independent and has been tested with the Cindi database that contains information on a virtual library.

1. Introduction

1.1 Natural language interfaces to databases (NLIDB)

One of the earliest and most widely studied areas of Natural Language Processing (NLP) is the development of a natural language interface to database systems (NLIDB) [8]. Such front ends relieve the users of the need to know the structure of the database and offer a much more convenient and flexible interaction. Because of this, this application of NLP is still widely popular today [12].

1.2 Question-Answering (QA)

Compared to NLIDB, open-domain question-answering (QA) is a fairly recent field in NLP and the approaches typically used in this domain can be very useful in NLIDB [6]. Recent developments in QA [13] have made it possible for users to ask a fact-based question in natural language (ex. *Who was the Prime Minister of Canada in 1873?*) and receive a specific answer (*Alexander Mackenzie*) rather than an entire document where the users must further search for the specific answer themselves [5]. In this respect, QA can be seen as the next generation of handy tools to search huge text collections such as the Internet [8][9].

1.3 NLIDB versus QA

Natural language interfaces to database systems and question answering systems differ fundamentally in their scientific goals and their technical constraints. QA systems try to answer a NL question by analysing a collection of unrestricted and unstructured texts; while NL interfaces to DB have the advantage of dealing with structured texts, that is, texts that have already been decomposed semantically; entities and relations have already been identified and related. However, both NL interfaces to DB and QA systems share an interesting similarity: they both take as input a question formulated in natural language and must interpret it in order to answer it properly. The advantage of the QA domain is that standard metrics do exist to evaluate and compare different approaches. In 2001, the best scoring system at the TREC conference used patterns and provided 50-60% correct hits when the answer was a passage of 50 bytes long [15].

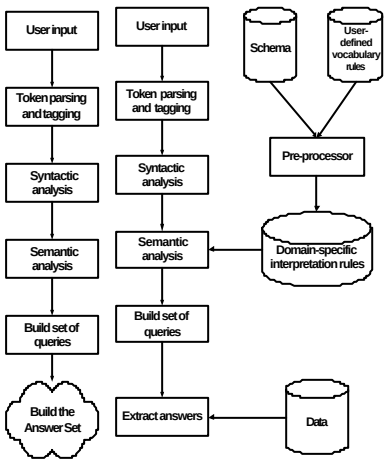

Figure 1. Left: The process flow of a typical NL interface to a data base Right: The process flow of our NL interface to the data base. The semantic parsing makes use of the schema.

Analysing an input question in QA and in NLIDB systems is mainly based on a part-of-speech tagging of the words (or tokens), often followed by a syntactic analysis (partial or full) and finally, a more or less precise semantic interpretation. Although there exist standard techniques for part-of-speech tagging and syntactic analysis, techniques for semantic parsing are still very varied and ad-hoc. In an open-domain situation, where the user can ask questions on any domain, this task is often very tentative and relies mainly on lexical semantics only. However, when the discourse domain is limited (as is the case in NLIDB), the interpretation of a question becomes easier as the space of possible meanings is smaller, and specific templates can be used [14].

We believe that meta-knowledge of the database, namely the schema of the database, can be used as an additional resource to better interpret the question in a limited domain.

The two strategies described above are presented in Figure 1. The left part of Figure 1 illustrates the general architecture of a NL interface system. The processing is linear,

going through the stages detailed above. The right part of the Figure 1 shows how semantic interpretation can be improved by making use of the meta-information, i.e. the schema of the database.

Once the question has been analyzed, the system tries to match the template representation to the knowledge base created by the pre-processor and to generate the SQL query.

2. Our Proposed Architecture

The detailed design of the NLP processor is presented in Figure 2.

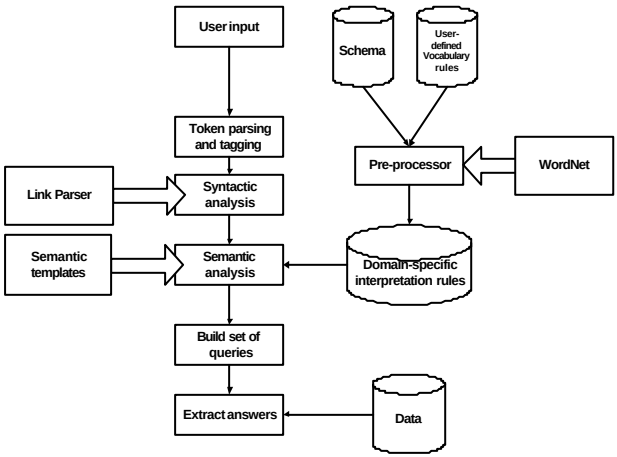

Figure 2. The design of our NL Processor

2.1 Semantic Sets

In order for the interface to take into account lexical variations in the input questions, the pre-processor builds a semantic knowledge base composed of interpretation rules and semantic sets for all possible table and relation names in the database. This allows us to build the same semantic representation, and hence, the same SQL query for questions like:

1. Who is the *author* of the book "Algorithms"?
2. Who is the *writer* of the book "Algorithms"?

To build the semantic knowledge base, the pre-processor first reads the schema of the database, and then identifies all table and attribute names and finally, it uses WordNet [7] to create a list of hyponyms, hypernyms and synonyms for each table and attribute name.

2.2 User-Defined Vocabulary Rules

While the semantic sets are built semi-automatically, certain vocabulary words and default rules can only be built by the system administrator. These include what we call action verb rules, default attributes rules and relation rules.

Action Verbs

To indicate the possible action verbs that can relate entities in a specific database schema, the database administrator should provide a list of action verbs and associate them with the table names. For example, Figure 3 shows two action verbs (*download* and *writes*) and how they can be used to relate to the Cindi dababase [1].

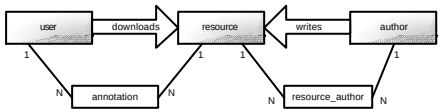

Figure 3. Two action verbs for the Cindi databse

Default Attributes

Default attributes are used to indicate which attribute is being asked for, when the user input does not contain an explicit reference to an attribute name. For example, in *Who wrote The Old Man and The Sea?*, the action verb *wrote* relates the tables *author* and *resource*, but the attributes of interest are not explicitly stated in the query as in *What is the name of the author who wrote the book with the title: The Old Man and The Sea?*. In order to determine which attributes are referenced when none are explicated, the database administrator must designate a default attribute for each table in the database.

Relations Rules

Relations rules are meant to take into account dependencies between relations; and indicate the general pattern of mapping a user input into an SQL query. For example, given the 3-table relation shown in Figure 4, if the tables *resource* and *author* are involved in a SQL query, then the table *resource_author* must appear as well. The relation rules determine the list of tables and the list of conditions in the final SQL query.

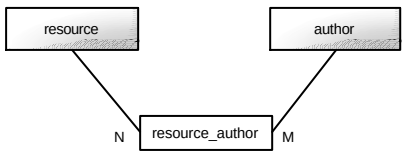

Figure 4. Example of a three table relation

The rule associated with the table configuration of Figure 4 is:

```
if (table_list contains (author, resource))
then table_list = table_list + resource_author
```

Each relation rule is associated to an SQL template, that created by the pre-processor, and used at run time to build the final SQL query with the specific values found the user input. For example, with the relation of Figure 4, the associated SQL template will be:

```
SELECT <attribute_list>
FROM author, resource, resource_author
WHERE author.author_id=resource_author.author_id
AND resource.resouce_id=resource_author.resource_id
AND <conditions_list>
```

2.3 Question Analysis through Semantic Templates

The analysis of the input questions is performed by matching the syntactic parse of the question to a set of fixed templates. This approach is similar to that of much work in QA [10]. Once we have taken into account the semantics of the individual words, we try to account for the interpretation of the entire question. For this, we use 3 main types of semantic templates:

<attribute> of <object>
<attribute> of <object> of <object>
<action verb> <object> <attribute value>

To determine that a natural language question matches a particular template, the question is parsed through the Link Parser [11]. As this parser finds a ranked list of parse trees (from the most likely to the least likely) for a given input, we try all parse trees in the order presented until one matches a template. As objects are mostly realized linguistically as noun phrases, to match an <object> in a template, we specifically look for the tags of the link parser that correspond to noun phrases. To match an <attribute>, any string will be considered, as long as it forms a full constituent. Let us now show each template in detail.

Attribute of Object of Object Template

This template is meant to take into account questions that involve one explicit attribute from a table and the default attribute from a second table. This is shown in Figure 5.

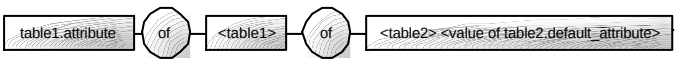

Figure 5. The attribute-of-object-of-object template

The associated SQL template is:

```
SELECT table1.attribute
FROM table1,table2 WHERE table2.def_att=<value of table2.def_att>
```

Action Verb Template

The action verb template is shown in Figure 6.

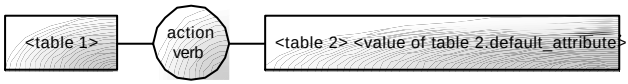

Figure 6. The action verb template

The simplest example for the action verb template involves one action verb and the value of one default attribute, as in *Who wrote the Old Man and The Sea?* The system does not find any table name in the input sentence, but it finds the action verb *wrote*, which is related to the tables *author* and *resource* as we already saw in the Relation Rules. From the default attribute table:

```
Attribute1=Table1.default_attribute=author.name
Attribute2=Table2.default_attribute=resource.title=`the  old  man
and the sea'
```

From the schema rules, *(author, resource)* involves *(resource_author)* as well. The final SQL is:

```
SELECT author.name
FROM author,resource,resource_author
WHERE resource.title='the old man and the sea'
AND resource_author.resource_id=resource.resource_id
AND resource_author.author_id=author.author_id
```

3. Implementation

In order to test our approach, we incorporated it within the Cindy database system. The Cindi system [2][3] is a virtual library system meant to increase the search results by using an appropriate index entry called the Semantic Header [4] and providing a mechanism to register, manage and search a bibliography. The accuracy of the system is currently being tested on the database for the Cindi library system. The schema contains 15 tables (relations). The natural language input varies from 3 to 20 words. By creating additional rules, and by improving the mechanism of generating the rules, we intend to raise the capabilities of the NLP system beyond these limits

At the present time, we estimate a success rate of about 70%[1] for questions having one action verb, one direct object and two modifiers. With three modifiers, the success rate decreases significantly because the input question becomes syntactically more complex and the number of possible parses found by the Link Parser increases exponentially.

[1] That is, 70% of the input questions have been correctly translated into SQL.

4. Conclusion And Future Work

If the data is highly organised, as is the case in the RDBMS area, the search domain can be reduced, and thus the precision of the question analysis can be improved. The reduction of the search domain is made possible because the system knows in advance, through the interpretation rules, what it can answer.

We plan to associate probability measures to the constructed SQL queries using our confidence level in the lexical and semantic relations used by the pre-processor and to the analysis of the question.

References

[1] Desai, B.C. (1997) Supporting Discovery in Virtual Libraries, *Journal of the American Society of Information Science*, 48-3, pp. 190-204.

[2] Desai, B.C., Shinghal, R., Shyan, N. and Zhou, Y. (1999) CINDI: A System for Cataloguing, Searching, and Annotating Electronic Documents in Digital Libraries, *Proceedings of ISMIS'99*, Springer-Verlag, Warsaw, Poland, June, pp. 154-162.

[3] Desai, B.C., Shinghal, R., Shyan, N. and Zhou, Y. (1999) CINDI: A System for Cataloguing, Searching, and Annotating Electronic Documents in Digital Libraries, *Library Trends*, Summer 1999, 48(1), pp209-233.

[4] Desai, B.C., Haddad, S.S. and Ali, A. (2000) Automatic Semantic Header Generator, *Proceedings of ISMIS'2000*, Springer-Verlag, Charlotte, NC, pp.444-452.

[5] Desai, B.C. (2002) Search and Discovery on the Web, *Proceedings of SSGRR'02*, July, L'Aquilla, Italy.

[6] Lehnert,W. (1978) *The Process of Question-Answering*, New Jersey.

[7] Miller, G. (1995) WordNet: a Lexical Database for English, *Communications of the ACM*, 38 (1), November, pp. 39-41.

[8] McTear, M. (1987). *The Articulate Computer.* Oxford, England: Basil Blackwell.

[9] Molla, D., Berri, J. and Hess, M. (1998) A Real World Implementation of Answer Extraction, In: *Proceedings of DEXA Workshop*, pp. 143-148.

[10] Plamondon, L. and Kosseim, L. (2002) QUANTUM: A Function-Based Question Answering System. . In*: Proceedings of The Fifteenth Canadian Conference on Artificial Intelligence (AI 2002)***.** R. Cohen and B. Spencer (Eds.), Lecture Notes in Artificial Intelligence no. 2338, pp 281-292, Springer-Verlag. Berlin. May 2002, Calgary, Canada.

[11] Sleator D., Davy Temperley, D. (1993) Parsing English with A Link Grammar. *Proceedings of the Third Annual Workshop on Parsing Technologies*.

[12] *TREC-9 (2000). Proceedings of the Ninth Text Retrieval Conference*, Gaithersburg, Maryland, USA (available at http://trec.nist.gov/pubs/trec9/t9_proceedings.html)

[13] *TREC-X (2001). Proceedings of theTenth Text Retrieval Conference*, Gaithersburg, Maryland, USA (available at http://trec.nist.gov/pubs/trec10/t10_proceedings.html)

[14] Watson, M. NLBean(tm) version 4: a natural language interface to databases! www.markwatson.com. Site visited in May 2002.

[15] M.M. Soubbotin, S.M. Soubbotin (TREC-X 2001) Patterns of Potential Answer Expressions as Clues to the Right Answers, see reference *TREC-X (2001)*