Signar: Generating responses in a natural language query system

H. Khelalfa, O. Nouali, Laboratoire L.R.D.I.
Rue des 3 frères Aissiou, Ben Aknoun, Alger (ALGERIE).
Fax: 213 (2) 79 21 26, Tel: 213 (2) 79 18 21

Abstract: Sigar is a natural language query system. With Sigar, a user may query relational databases in natural language, use implicit queries while conversing with the system, and obtains cooperative responses owing to the use of pragmatic knowledge and integrity constraints. Sigar modular structure makes it portable and easy to adapt to new applications. This paper presents Sigar main feature: the generation of informative responses.

Key words: Natural language interface, natural language understanding, responses generation, cooperative responses.

1. Introduction

The designer of a natural language database interface is confronted to the problems of understanding users’ queries and generating responses in natural language. In order to produce appropriate answers, cooperative systems should not be able to respond merely to user’s queries by translating the results returned by the database management system (DBMS), but they should be able to make inferences from data.

In practice, most existing natural language systems accept natural language questions, but may produce indefinite, irrelevant and wrong responses [12]. These responses can arise, for example, when one or more of a user’s assumptions are incorrect.

For instance, let’s assume that user U examines a database of the courses offered by University X.

U is interested in information on students whose math score is higher than 10. The following dialogue is obtained:

User: How many students scored higher than 10 in math during the 1989 fall semester? System: none.

[The user then, asks a more general question]
User: Anyone failed math, during the 1989 fall semester?
System: no.

[The user asks again the question differently]
User: How many students passed math during the 1989 fall semester?
System: zero.

[The user concludes from this dialogue that, no one has failed, and nor passed math. Finally, the user understands.]

2. Architecture & Overview of SIGAR

Once the query in natural language is entered, a preliminary parser filters it out to suppress expressions that are not necessarily needed in understanding and interpreting the query (for example, politeness expressions). Then, this new form of the query is divided by the “Morpho-Lexical” parser into lexical entities. Using an application dependent lexicon, this parser extracts useful information for
each entity. Then, the entities and their definitions are passed to a “Syntactical-Semantic” parser that triggers an ATN interpreter. The query is then checked against the ATN grammar. This will yield:
- The intermediate representation, built progressively during the analysis, which will be used to translate the query into a database query language.
- The different presuppositions that the user has implied. These presuppositions are explored in order to generate appropriate answers when necessary.
Finally, the response generator will provide the user with a cooperative response in natural language. In the case of an implicit query, the generator calls an ellipsis module which consult the dialogue history to make it explicit.

3. Responses Generator

This section defines the concepts of direct and indirect response, introduces Grice’s[12] principles of cooperation, and describes the general algorithm used by the response generator along with the different types of knowledge it uses.

3.1 Definitions
- Direct response: A direct response is a response delivered by the system, which displays database responses without additional information.
- Indirect response: An indirect response is an informative response generated by the system when there is no positive response in the database (null response) to the user’s query.
- Cooperation principles: Grice elaborates three basic cooperation principles which are accepted as model to build natural language interfaces[12].
  * The maxim of quality: do not make a contribution which one believes to be false or which adequate evidence is lacking;
  * The maxim of quantity: be as informative as required, but no more;
  * The maxim of manner: avoid obscurity of expression, avoid ambiguity, be brief (concise).

3.2 Description of the responses generator

In general, a user’s query carries several presuppositions. One of the underlying principles of Sigar is the analysis of all the assumptions extracted from the user’s query in order to determine the type of response to produce i.e., direct or indirect. The presuppositions are recognized during the query analysis phase using inference rules.
Example:

  Question: Display the firstname and the address of the technician whose lastname is “ALILOU”.

In this question, the user assumes that:
- Alilou is an existing technician.
- A technician has a lastname, firstname and adress, that is, the existence of the
semantic relationship between entities “technician”, “lastname”, “firstname” and “adress”.

### 3.2.1 Positive responses

The system formulates a natural language response from the results returned by the database, by translating eventually all the codifications and using various messages.

**Example:**

**Question 1:** Display the manager of each department.
**Response 1:**

<table>
<thead>
<tr>
<th>Manager</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali</td>
<td>Electronics Engineering</td>
</tr>
<tr>
<td>Halim</td>
<td>Manufacturing</td>
</tr>
</tbody>
</table>

**Question 2:** Display the degree for the scientific “Doudou”.
**Response 2:** The degree of the scientific “Doudou” is PhD.

### 3.2.2 Null responses

Sigar processing of null responses is not based on the negation principle, i.e., the absence of a fact in database, supposes that the negation of the fact is true”. This maxim isn’t always true. It may lead the user to erroneous interpretations of a response. The system, therefore, violates the maxim of quality outlined above.

**Example:**

**Question:** Does the technician “Tata” supervise the “Snark” project?
**Response R1** below is simply a direct answer.
**R1:** No

R1 may mislead the user. For instance, he/she may be led to think that although technician “Tata” does not supervise the Snark project, “Tata” may supervise other projects, when in fact a technician cannot be a project supervisor. A more appropriate answer would be “no, a technician cannot be a project supervisor “. Therefore, the system corrects all of the user’s misinterpretations.

Sigar interpretation of null response is based on Motro’s[26] and Janas[15]. From the formal representation of the user’s failed query, the system recursively modifies and generalizes the failed query until no more failed queries are found. This approach requires a set of rules to modify the initial representation of the query, and an algorithm using these rules to generate responses.

### 3.3 Ellipsis Forms

An Ellipsis is an omission of one or several words in a sentence without altering the meaning of the sentence. Ellipsis forms are taken into account by the generator. So, that the user can converse with the system through the use of implicit queries. During a user’s session, a history of the dialogue is kept in the form of graphs. The nodes represent the user request and the arcs represent a chronological link between requests. When the generator detects an omission, it consults the history to make the query explicit.
Example:

User: what is the degree of the scientist Moussa?
System: PhD.
User: And his address?
System: 15 av. colonnel Amirouche, Alger.
User: And Boussad’s?

A new graph is created, and the old one is destroyed, whenever a new explicit query is formulated.
An initial node is first created, and each time an implicit request is expressed another node is added and linked to the previous one.

3.4 Response Generator Algorithm
The generator checks all the user’s presuppositions recognized during the query analysis phase.
(i) if all of the user’s presuppositions are true and at least one integrity constraint satisfies the required data, a indirect response is generated using this constraint without any accesses to database.
(ii) if one of the user’s presuppositions is false, a indirect response is generated using this presupposition as explanation of the failed query.
(iii) if all of the user’s presuppositions are true or null, the response is generated after acces to the DBMS.
- in case the response isn’t null, the generator produces a surface language version of the response.
- if the query returns a null response, the generator attempt to generate a general response. The query is generalized by removing or substituting a condition in the search criteria.
- if the generator doesn’t retrieve the general response for the user’s query, it checks the null presuppositions to generate indirect response.

3.5 Generator knowledge
Constructing an adequate response requires the use of several types of knowledge: inference rules, knowledge about the database structure, integrity constraints and pragmatic knowledge.

3.5.1 Inference rules
The generator uses inference rules in order to produce appropriate responses when the DBMS response is null.
Example:
The question “Can Siln project begin next month?” does not have a positive response, because “Siln” project does not exist in the base.
The following rule: “starting a project → existence of the project”, allows to get an indirect response, as: “No, Siln project is not defined, so it can not start”.


3.5.2 Knowledge about database structure
The generator must produce more appropriate response using knowledge about the database structure, since querying a database using a natural language does not call for the user to be familiar with the database structure.
Example:
Question: what is the telephone number of Sonatrach company?
Response: There is no information about the telephone number of companies.

3.5.3 Integrity constraints
The generator also uses information present in database in the form of integrity constraints to correct some of the user's false assumptions. When a question is formulated several situations may arise:

a) The question may violate an integrity constraint. In this case, it is not necessary to access the base.
Example:
Suppose the constraint (i): "The project supervisor must be a scientist"[23].
In the question Q1: "Display the projects supervised by technicians, during the year 1987", the user assumes that a technician can be a project supervisor. There is no positive response to Q1 because the constraint (i) require that a project supervisor must be a scientist, not a technician. The response R1: "No one", is literally correct. However it suggests that the techniciens do not supervise projects for the year 1987 only. On the other hand, the response R2: "A technicien does not supervise a project", corrects the user's knowledge about the data base.

b) A relationship exists between the integrity constraint and the question.
- A data base access is not necessary: the response is directly deduced from the integrity constraint.
Example:
Question: "Which projects are supervised by scientists ?".
Response: "All the projects are supervised by scientists".
- A data base access is necessary.
Example:
To the question: "which projects are supervised by the scientist Moussa ?", a response may be delivered after an access to the data base to verify the existence of the value "Moussa".

3.5.4 Pragmatic knowledge
Another type of knowledge assists the generator process to build informative responses. This knowledge is called pragmatic knowledge. The context is used:
- to resolve ambiguities
- to extend the response by completing the formal representation, whenever possi-
ble. So that responses become non ambiguous and respect Grice’s principles. The idea is to:
- Avoid responses based on a literal interpretation of the questions.

Example:
The response to the question: “Can I have the number of institutions in Algiers?” must be an informative response and not a response as just “Yes” or “No”.
- Infer default value, that is, to deduce information not explicitely included in the user’s query in order to produce informative response.

Example:
The following query: “I would like to know all the Algerian scientists” does not specify which information about the scientists the user wants to know (last name, first name, ...). The generator adds, by default, the attribute last name (which is the most characteristic attribute of the relation scientists) to the formal representation. So the response will be: “The names of Algerian scientists are: Omar, Souad, ...”.

4. Administrator

The SIGAR administrator is a tool addressing administration and maintenance issues. The current services provided by the SIGAR Administrator are:
- An automated generation of an application dictionary.
- An easy modification of an application dictionary.
- An easy extension of SIGAR grammar.

Hence the DBA’s tasks is alleviated from the burdens of application portability, dictionary update, changes in users’ query formulation.

5. Conclusion

Sigar is a natural language query system. It allows, a user to query a relational database in natural language, and to obtain cooperative responses owing to the use of pragmatic knowledge and integrity constraints.

In addition, Sigar makes a user’s interaction with the machine more humanlike by allowing ellipsis. The SIGAR Administrator alleviates the DBA from cumbersome and tedious tasks. This paper presented Sigar main feature, the generation of informative responses. An informative response is given by:
- Completing the meaning representation obtained at the analysis level, using pragmatic knowledge,
- translating eventually all codifications and using messages to make the response returned by the database understandable by the user,
- giving indirect responses by using a set of knowledge as the integrity constraints, to correct some user’s false assumptions.
References


[3]: Alain COULON, Informatique et langage naturel. technique et serie en informatique.volumc 5, n 2, 1986.

[4]: L. DANLOS, Generation Automatique de textes en langues naturelles, Masson 85.


[12]: Annie GAL, Des reponses cooperatives donnees par un interface SGBD en langage naturel, 5eme congres: Reconnaissance des formes et Intelligence Artificielle, tome 1, Grenoble, Novembre 1985.


[18]: Jean Henri JAYEZ, Outils informatiques pour le traitement de la langue fran_aise, Colloque, informatique et langue naturelle, Nantes, Octobre 1988.


[23]: Ounissa LARAB, Systeme de Gestion du Potentiel Scientifique et Technique National (PSTN), CERIST, Decembre 89.


[28]: W.A. PERKINS, Generation of natural language from information in a frame structure, Data & knowledge Engineering, April 1989, pp 101-114.
