MODELLING HUMAN-COMPUTER INTERACTIONS

IN A FRIENDLY INTERFACE

Patrick SAINT-DIZIER
UNIVERSITE DE RENNES I
I.R.I.S.A.
Campus de beaulieu
35042 RENNES FRANCE

A B S T R A C T

This paper describes a way of building an intelligent interface between a human and a computer. We first examine the main characteristics of such a system from three points of view: the user, the expert in the definition of applications and the computer scientist. Our job is to find an appropriate formalism that can describe the different aspects of our system: natural language understanding, knowledge representation, explanation and control mechanisms, plan generation, relational databases and meta-knowledge representation. Finally, we conclude by pointing out some remaining problems that will require additionnal basic research.

The job described here has been implemented in PROLOG computing language on the CII HB 68 of the IRISA. Our assistant interfaces an application, called Cigare, whose role is to help people in scheduling meetings.

Keywords: Office automation, knowledge representation, natural language, relational databases and cognitive modeling.

1_ INTRODUCTION :

The close interactions between the fields of computer science and cognitive psychology have given rise to what is generally known as human information processing models of cognitive processes. In this paper, we describe an intelligent interface between a human and a specialized computer (such as: text-editor, interactive query system, electronic-mail, ...). We call this interface a user assistant. Its role is to be an expert of the functions of the application to which it is connected and to provide a friendly environment to people who wish to use this application. In this job, we look on our assistant from three points of view:

-the user's point of view. He uses the application via the interface. Very often, he is not a computer professional.

-the computer scientist's point of view. Its role is to build a man-machine interface adaptable to various kinds of applications.

-the expert's point of view. An expert is a specialist in the definition of applications. His task is to specify to the assistant the different parameters of the application he wants to interface.

In the next section, our purpose is to gather some of the most interesting characteristics a human-computer interface must have, from the user's point of view. In section 3, we take up the expert position and we describe what structures are necessary so as to enable the expert to describe properly the different aspects of the application he wants to interface. Finally, in section 4, we describe the main problems the computer scientist is confronted to.

This job is an attempt to formalize and to solve some of the problems we have met in our investigations. We don't claim to solve all the difficulties and we think that a lot of basic research remains to be done.

2_ A FRIENDLY INTERFACE FOR USERS

With the help of psychologists and application builders, we have brought out some of the main properties an intelligent interface must have [11]:

2.1_ Understanding the User's Requests:

Interacting with a computer in a "natural" way is much more perceived as "user friendly". In fact, it will be always necessary for users to make the effort of learning what a system is capable of doing, but natural language would minimize the efforts of learning how to make the system do it. In addition, natural language allows a casual user to use more advanced capabilities of a system without knowing the exact commands. But, as yet, due to the vast amount of information to store, the domain of the discourse has to be highly restricted. The assistant has, in fact, to know far more than the syntactic rules that allow translating natural language into a formal representation (cf. section 4). In order to avoid frustating the user, the domain of the discourse has to be carefully studied for the assistant can understand immediately most of the inputs and the rest after one or two rephrasings.

In other respects, some experiences have shown that when users have some difficulties to express a request (especially a request for help), the best solution is that the assistant provides them with a menu driven dialog. However, this menu driven dialog has to be carefully studied in order to take into account the main problems a user can meet.

When the user expresses himself in a "natural" way, most of his requests will not correspond exactly to the input forms of the service actions. The user assistant must therefore have the capability of

mapping a given request into the appropriate commands of the service that fulfil that request.

It is also important that the assistant generates a paraphrase of what it has understood in a request and that it waits for an acknowledgement from its user before it sends a command to the application.

2.2_ Responding Intelligently to the User's Inquiries:

In order to be really friendly and helpful, the assistant must be able to answer questions about:

-informationsthe user has already submitted,

-the current state of the application;

-the linguistic competence of the assistant (words or grammatical structures it knows). The user may ask for exemples of sentences the assistant can parse.

-how to perform a given task,

-which tasks the user may perform at that precise moment,

-why he can't perform a task,

-why and how the assistant has made some deductions (ex: how it has solved unknown references).

In the answer of the assistant, a recall of the question has to be mentionned so as to ensure the user that the assistant has understood properly his question.

2.3_ Detecting User's Failures and Making Explicit the System Limits:

Both the user and the assistant may fail for various reasons. We differentiate two types of failures: input and model failures. Input failures are due on the one hand to grammatical and semantic mistakes from the user and on the other hand to linguistic structures the assistant doesn't know and to misperception of a word when using vocal terminals. Model failures are due to the user's ignorance or bad understanding of some aspects of the application.

The assistant has to detect these failures and to provide explanations to its user. Concerning input failures, the assistant has to point out unknown words, semantic incompatibilities and sentence structures it can't parse. When it meets an unknown word in a sentence, it must try to deduce the meaning of this word from the remainder of the sentence. If it succeeds, it has then to ask the user for a confirmation. Concerning model failures, it must point out false presuppositions, incomplete requests and unknown actions to the application. It must be really informative (but not too talkative !), showing clearly and explicitly why it cannot accept a request. It must provide explanations, exemples and alternatives or restatements.

2.4_ Acquiring knowledge :

The user assistant is particular to a user. Consequently, it must be adaptable to his sensibility and habits. It must be able to learn some new information, so as:

-to increase its linguistic competence. Our assistant can learn new synonyms of words that it already knows. The user as just to declare: "X is a synonym of Y". We think that the acquisition of new words and new grammatical structures has to be done by a human expert in linguistics because most of the users have not the required competence to perform this task.

-to take into account behaviour specifications, in order to avoid disturbance to its user. The user may give instructions to its assistant, such as "My meetings always take place in room no 210.". They play the role of default options.

3_ ROLE OF THE EXPERT

The assistant, application independant, is built by a computer scientist. To interface a given application, some parameters of the assistant have then to be instantiated. This is the role of the expert. The two main classes of parameters are the linguistic parameters and the parameters that describe the functions of the application. The specification of these parameters is done via a specific language.

3.1_ The Linguistic Parameters:

In the previous section, we have explained that the domain of the discourse has to be restricted. Consequently, it is not possible to store in our assistant, once for ever, all the vocabulary of a given language. That means that the expert will have to find , for each application to interface, all the required vocabulary so as to allow the user to express himself in natural language with sufficiently various expressions. Some words, such as articles and prepositions are common to all the applications but most of the words are application dependant. In order to limit the job of the user, it is useful to implement in the assistant, once for ever, a set of rules that describe the various morphologies (plural, feminine, conjugation forms ...) of any given word. So, the expert has only to specify the infinitive form of verbs, the masculin singular of adjectives, etc... For each of these words, the expert has to give:

-the syntactic category of the word (noun, verb ...),

-if this word accepts complements.

This last point leads us to introduce semantic features so as to enable the expert to precise what kind of complement is acceptable. As we are concerned by a small subset of natural language, it is possible to define semantic categories in a finite number and to include each word in, at least, one of these categories. We think that these categories are limited to the set of categories of objects on which the application operates (human, time, place ...). Finally, the structure of a lexical

item is composed of :

-a word,

-a syntactic feature (noun, verb,...),

-a semantic feature, linked to the word itself (except for verbs where this feature is the feature of an acceptable subject),

-a list of semantic features of acceptable complements, with the preposition that introduce them.

Exemple:

WORD(assembly,noun,meeting,(of,human).(of,place).nil).

The rules that describe the grammatical structures are application independant. The main rules are implemented in the assistant once for ever. However, we think it is important to allow the expert to add new grammatical structures and descriptions of idiomatic expressions. This can be done via a specific language [6].

3.2_ The Description Of The Application:

The expert describes a model of the application to the assistant. The first goal of this description is to make the assistant "understand" the kind of request the user has submitted. The second role is to enable the assistant to help the user. The word "understand" means, here, to find the exact meaning of a request with regard to the functions of the application. It also means to verify if this request is possible considering the previous actions the user has performed and the data transmitted by the application.

To model an application, the first task is to decompose it into basic actions. An action will be identified by a set of significant patterns to find in the output form produced from the user's request. Next, it is necessary to describe the conditions under which an action may be performed. These conditions express that, previously, some actions must have (or must not have) been performed by the user and (or) some information must have (or must not have) been transmitted by the application. These information are the result of the user's actions or the result of other user's actions in the case of multi-users applications. Finally, for each action, an exhaustive list of tasks the assistant has to do is described in terms of information to add to a contextual database and commands to send to the application.A rule that describes a basic action of an application is of the form :

RULE(<identification>,
<list of patterns to find in the request>,
<conditions >,
<information to add to the contextual database>.
<commands to send to the application>).

4_ ARCHITECTURE OF THE SYSTEM : THE COMPUTER SCIENTIST POINT OF VIEW.

The job of the computer scientist is to build a system that takes

into account both the different parameters specified by the expert and the human-computer interactions constraints we have point out in section 2. In this section, we first examine the linguistic component and next see how the request is interpreted. Finally, we explain how the assistant can provide help to its user.

We first assume that the human and the machine interact in a way that can be described by a production system. Indeed, we think that the formalism of production systems and logic is a good formalism that has been really successful in providing insights in both theoretical and practical aspects of computing science. Then, we have to specify:

-A general structure to describe the rules of this production system,

-the process by which rules are selected for execution, and how to express it,

-the structure of the information utilized by the rules,

-how the information reflects the current state of the knowledge on which the system operates,

-the operations on the rules (modifications, ...).

Figure 1 shows the overall structure of a user assistant:

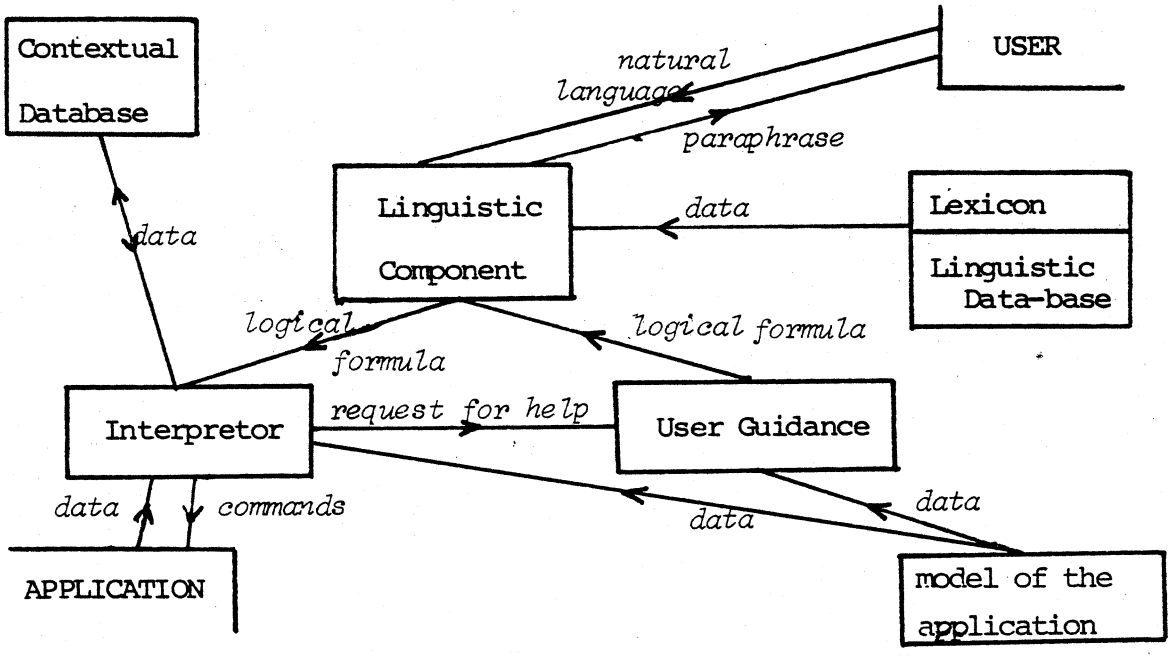

fig. 1

4.1 The Linguistic Component:

The goal of the linguistic component is to generate a formal representation of the meaning of the natural language sentences of the user. Actually, there exist many various ways to represent formally a

natural language sentence. We think that the formalism adopted and described by [6], [8] and [17] is very well adapted to our problem. In this formalism, the study of determiners have been looked at indetail so as to refine the range of quantifiers. In addition, some tools have been added so as to represent better some structures such as questions begining by : Why, How many, How much and the expression of time. This representation is in higher level logical form. For instance, the question

"Who are the participants of the meeting A ?"
has the following formal representation :
QUESTION(SET-OF(x),. meeting (A).participant-of(x,A))

Our parser is composed of two entities: a lexicon and a set of rules that describe the grammatical structures of french. We think that it is important that these rules may also be applied backward, for sentence synthesis. In fact, a lot of job remains to be done about this problem. The main problems we are confronted to about sentence synthesis is that we must specify all the syntactic constraints, so as to have a correct output, and to ensure ourselves that all the information the logical form contains has been synthetised in a correct way.

In our system, a sentence is parsed by a grammar where:

(1) The axioms are a finite set of modes : declarative, imperative....

(2) The non-terminal symbols (SN, SV, Verb ...) have the general form:
X(<syntactic features>,<semantic feature>,<formal representation>)

The syntactic features (gender & number) and the semantic feature (human, place, ...) are the features that result of the parse of the sentence substructure represented by X.

(3) The terminal symbols, that are the lexical items.

(4) The rules, that have the form:
$X(l(SY_1,\ldots SY_i),g(SE_1,\ldots SE_i),h(F_1,\ldots F_i)) \longrightarrow$
$Y_1(SY_1,SE_1,F_1) \ldots\ldots Y_i(SY_i,SE_i,F_i)$

are applicable iff $s1(SY_1,\ldots SY_i)$ and $s2(SE_1,\ldots SE_i)$ are true. Where:

- SY stands for the syntactic features,
- SE stands for the semantic feature,
- F for the formal representation of a substructure of a sentence.

s1 and s2 are functions that express conditions, such as pattern-matching, between features that come from each non terminal symbol on the right part on the rule. l and g are functions that describe how to build the syntactic and semantic features of the non terminal symbol on the left part of the rule from the features on the right part of that rule. f describes how to build the formal representation of the sub-expression parsed by the rule. The main problem is that the meaning of a complex expression has to depend only on the meaning of its subexpressions. Every well formed subexpression is then considered as a unit of meaning that can be integrated in a larger

expression.

Finally, the lexicon and the rules of the parser are considered as a database so as to enable the system to give information about its linguistic competence (cf. the nice job referred in [14]). This structure also allows us to implement procedures that detect, in a simple way, grammatical mistakes and semantic inconsistencies.

4.2_ The contextual database :

We think that it is important that a request don't be treated as an isolated event. A context is built up so as the repeated exchanges between the user and his machine may be considered as approaching a simple but real conversation. An image of all the exchanges between the user and his assistant and between the assistant and the application is stored in a contextual database. The contextual database is local to a user and is composed of a list of facts that represent:

(1) The information the user has transmitted to the application via his assistant. This can be looked as an historical database.

(2) Informations about the state of the dialog between the user and his interface (what the user knows and what he is talking about).

(3) The information the application has transmitted.

(4) Some behaviour specifications, given by the user.

The contextual database plays the role of a short term memory. All the facts are represented in the same way :

FACT(<kind of fact>,<information>,<source of the information>).

The argument "source of the information" allows the assistant to explain, at any time, the origin of the information (inheritance or deduction, default option, the user's request).

4.3_ Interpretation Of The User's Request:

We have examine in section 3 the structure of the rules that describe an application. In our job, a rule is of the form:

Rule(<name>,C1,C2,L,T).

Where:

-C1 is a set of conditions on the existence of some facts, stored in the contextual database,

-C2 is a set of patterns to find in the logical form produced from the user's request. C2 allows an efficient preselection of rules and is a tool for user guidance (cf. 4.4),

-L is a list of information to collect in the logical form . Some control procedures, linked to the informations, are described here (cf. example),

-T is a list of actions to perform (commands to send to the application and information to store in the contextual database).

Let's look at an example :

```
RULE(positive_answer_to_an_invitation,
    facts(exist(meeting).to_be_invited(user,meeting).nil),
    patterns_to_find(agreement_to_come.nil),
```

```
to_collect(dates(day<=31 and 8<=time<=20).nil),
actions(addCD(positive_answer(user).dates(<dates>).nil).
smAPP(positive_answer(user,<dates>).nil).nil)).
```

addCD stands for "add to the contextual database"
smAPP stands for "send commands to the application"

After the parsing process of the user's request, the interpreter evaluates the C1 and C2 of each rule. Thus, a preselection of (one or more) applicable rules is done. Then, the interpretor asks the user for a confirmation of its understanding. If it is correct, L and T of these rules are executed. If it is not correct, the user has to say which rule is applicable, if any. The user may also ask for help.(cf. user guidance module).

4.4 User Guidance:

One of our main principles is to never left the user to himself. The assistant must be able to help the user at his request or when it detects some failures. User guidance is a very vast problem, let's look at some aspects of it:

(1) What is the linguistic corpus necessary to express requests for help? Is natural language well adapted?

(2) When the assistant is not able to answer a question, how to make it express the reasons why it cannot (instead of the laconic expression "I don't know")? How to make it propose alternatives?

(3) How to help the user to plan his work?

(4) How to learn to novices the main functions of an application?

(5) How to be really very informative, without excess? For instance, is it possible to define different levels of information?

Despite the current interest in user guidance, we think that the design of a helpful and informative interface remains to be done. However, some very interesting and valuable results have been obtained in some works such as : [2], [14], [17], ... In our job, the formalism of the rules that describe an application allows the assistant to provide some explanations:

- When the user doesn't know which actions he may perform at a precise moment, it is from the context and through the evaluation of C1 of all the rules that the assistant gives him the list of the actions he is allowed to perform.
- If the user doesn't know how to perform a given action , the description of the L of that rule gives him the amount of knowledge required to perform this action.
- The evaluation of the C1 and the addCD of T of all the rules allows the assistant to generate plans [9] and to propose to the user various chains of actions (or subgoals) to reach the requested goal. This is a way to learn to novices how to use the application.
- When the user wants to perform an action that is not allowed, the evaluation and the description of the C1 of that rules gives the reasons why this action is not possible, and under which conditions it would be possible.

Actually, the user is guided by a menu-driven language to formulate his request for help. Requests for help in natural language are, indeed , very difficult to express and to be interpreted.

5_ CONCLUSION .

We have presented here a human-computer interface which is to be used by a large and casual public. We have check off what must be the main properties of such an interface. The formalism adopted here, based upon logic, reveals itself to be quite robust and general. This interface has now to be tested by users.

However, a lot of job remains to be done, especially in the following areas:

*How to respond more intelligently to incorrect inputs and to questions about the knowledge.

*How to built an expert that is able of reasoning on incomplete knowledge.

*How to process some linguistic problems such as fuzzy expressions.

*How to build an efficient "expert" to manage knowledge acquisition.

The job we have presented here has been implemented in PROLOG on the CII-HB 68 of the IRISA. The implementation has lasted the equivalent of 14 monthes for one person, but some work remains to be done to increase the performances. The application that our assistant interfaces is called CIGARE, its role is to help people in scheduling meetings. We also intend to connect this interface to a text editor.

REFERENCES

[1] BOBROW D.G., KAPLAN R.M., KAY M., WINOGRAD T. GUS: a frame driven-dialog system. A.I. Vol 8 no 1 1977.

[2] KAPLAN S.J. Cooperative Responses from a portable Natural Language Query System. Artificial Intelligence n° 19 , 1982.

[3] DOYLE J. A glimpse on truth maintenance. Proc. of the IJCAI 79 .

[4] GROSS M. Methodes en syntaxe. Hermann. PARIS 1975.

[5] HIRSHMAN L., PUDER K Restriction Grammars in PROLOG. Logic programming conference, Marseille 1982.

[6] MCCORD M.C. Using slots and modifiers in logical grammars for natural language. Technical report no 69-80. Univ. of Kentucky, Lexington, 1982.

[7] MINSKY M. A framework for representing knowledge, in P. Winston(ed).

[8] MOORE R.C. Problems in logical form. SRI project report . April 1981.

[9] NILSSON N. Principles of artificial intelligence. Tioga Pub. Co 1980.

[10] NORMIER B. Le Systeme SAPHIR. ERLI. 1982.

[11] QUINIOU R., SAINT-DIZIER P. Man-machine interface for large public applications. IEEE Seminar Zurich March 1982.

[12] PEREIRA F. Extraposition Grammars. Logic programming workshop. Debrecen, Hungary 1980.

[13] REITER R., NASH-WEBER B. Anaphora and logical form: on formal meaning representation for natural language. 5th IJCAI.

[14] SABATIER P., PEREIRA L.M. ORBI: an expert system for environmental ressource evaluation through natural language. Logic Programming Conference, Marseille, 14-17 september 1982.
[15] SAINT-DIZIER P. Some aspects of knowledge representation in an intelligent Man-Machine interface. The Mind and Machine Congress, Middlesex polytechnic, London, April 1983.
[16] SHANK R.C., ABELSON R.P. Scripts, plans, goals and understanding. Lawrence Erlbaum Press, Hillsdale N.J. 1977.
[17] WARREN , PEREIRA F.: An efficient easily adaptable system for interpreting natural language. Research report 1981.
[18] BESNARD P., QUINIOU R., QUINTON P., SAINT-DIZIER P., TRILLING L. Dialogue et representation des information dans un systeme de messagerie intelligent. Research report no IRISA 185. January 1983.