An Intelligent Interface to a Database System*

Alfredo Fernandez-Valmayor (*) Carlos Villarubia (**) Manuel Buenaga (*)

(*) Dep. Informática - Fac. Físicas - Universidad Complutense de Madrid
28040 Madrid, Spain.
valmayor@fis.ucm.es - mbuenaga@dia.ucm.es
(**) Dep. Informática - E.U. Informática - Universidad de Castilla-La Mancha
13004 Ciudad Real, Spain.
carlos@eui.uucp

Abstract

In this work, we describe the architecture of an intelligent interface that improves the effectiveness of full text retrieval methods through the semantic interpretation of user's queries in natural language (NL). This interface comprises a user-expert module that integrates a dynamic model of human memory with a NL parser. This paper concentrates on the problem of the elaboration of index patterns out of specific cases or instances. The structure of the dynamic memory of cases and parsing techniques are also discussed.

Introduction.

Many researchers have focused on the study of human memory processes to find a model to improve the way information is stored and retrieved from computers (Foltz, 1991). They believe that better understanding of how human memory indexes and searches information would facilitate the development of new and more effective methods of building databases. Our approach deviates from this. We try to use this better understanding of how human memory works, not to build new and more efficient databases, but to build tools that will make possibly to use more effectively pre-existent ones. The basis for this improvement is the construction of an intelligent interface. An interface that integrates a model of human memory with a natural language (NL) parser.

In some database systems (DBS) the user, if he is not an experienced one, can choose to interact with the computer through a human intermediary, which helps him to find the information he wants. Interacting with a human expert has the additional advantage that the user does not need to learn any formal language; however, even with human intermediaries information retrieval of textual databases may have a very poor performance. As an example, Blair and Maron (1985) report on a precision index of 80% and a recall index of less than 20% in a legal information retrieval system.

Several researchers have discussed the possibility of integrating an expert module in a DBS to automate the role of the human expert (Borgman, Belkin, Croft, Lesk, & Landauer, 1988). This has been our approach. The interface we present here would eliminate in most cases the need for this human expert. In this interface a user-expert module improves the effectiveness of the retrieval process through the semantic interpretation of user's natural-language query formulations. For that reason, NL processing capabilities are at the core of the system. In the user-expert module NL processing capabilities are founded on memory based parsing techniques; an anticipated improvement of conceptual parsers (Riesbeck, 1986; Riesbeck & Schank, 1989).

In this paper, we begin with an analysis of the process of information retrieval. This will be followed by an overview of our system and a discussion of the main issues motivated by the development of the prototype: the knowledge representation and knowledge acquisition techniques used to model expertise; the structure and indexing of cases in the memory model and finally the memory based parsing techniques based on this model.

The system we have had in mind when designing our prototype is a full-text retrieval system. A system designed to search a file of unstructured, or partially unstructured, textual information (Fernández-Valmayor & Villarrubia, 1991). Information retrieval

* **Acknowledgment**: This work is supported in part by CICYT grant TIC91-0217-C02-02: "Arquitectura para Interfaces en Lenguaje Natural con Modelado de Usuario" and CICYT grant TIC92-9958: "Metodología de diseño de herramientas de desarrollo para sistemas expertos basados en casos"

(IR) from this kind of system is to some extent more complex than retrieval from structured databases, since retrieval cues can be matched by many words in the documents.

A variety of methods exist to recover information from textual databases (Salton & McGill, 1983). Boolean expressions of terms used in the documents are the most simple approach. Other more sophisticated methods associate vectors of weighted terms with documents (Salton, 1989). In these later methods, documents are represented as vectors with weighted components. User's queries are also represented as vectors in the same space, therefore distance between vectors representing queries, and documents can be computed. The documents retrieved would be those with a distance less than a threshold quantity as specified by the user.

The effectiveness of a retrieval system can be evaluated in terms of the so called recall and precision indexes. Recall index measures the proportion of relevant documents out of all relevant documents in the database, while precision index measures the proportion of the recovered documents that are found to be relevant to the user's needs. In practice, recall and precision indexes tend to vary inversely; i.e., very specific query formulations retrieve few non-relevant items but also relatively few relevant ones. Query formulations can be altered to reach the desired recall and precision levels, using recall-enhancing devices; e.g., term truncation, and precision-enhancing devices; e.g., term weighting (Salton, 1986).

System overview

The system we are working on is an interface to existing databases. Accordingly, in designing our system, we mainly deal with the the strategic elaboration and verification stages of the information retrieval process. In the first stage, the user can express his information needs in natural language. The user-expert module will translate these input queries into a network of conceptual dependency structures (CDs) that later it will use to generate optimized retrieval cues and to make the mapping into the specific query language of the accessed database. Once the user has queried the system and retrieved the information, he can inform to the system on the appropriateness of the retrieved documents.

The two main components of the user-expert module are a dynamic knowledge base and a NL parser. We conceive this knowledge base as a model of human memory that incorporates knowledge about the database's domain and about the context of individual users.

The way in which the user-expert module lays between the users and the DBS is depicted in the diagram. From the diagram, we can see that if the user is an expert he can interact directly with the DBS. If the user is not an expert, then he can express his questions in natural language. The NL interface parses the user query building a conceptual dependency structure (CD) that captures the meaning of the input.

Parsing of user's input is based on prior memory structures. In the knowledge base, the long term memory has at its top level the dictionary used by the conceptual parser (Dyer, 1983; Birnbaum & Selfridge, 1981). The short term memory, also in the knowledge base, is a list of CD structures under construction that represents the user-computer interaction during the session. The CD structures in the short term memory represent both the meaning of the input already seen, and the expectations that will guide the parser to translate next input and to optimize the queries.

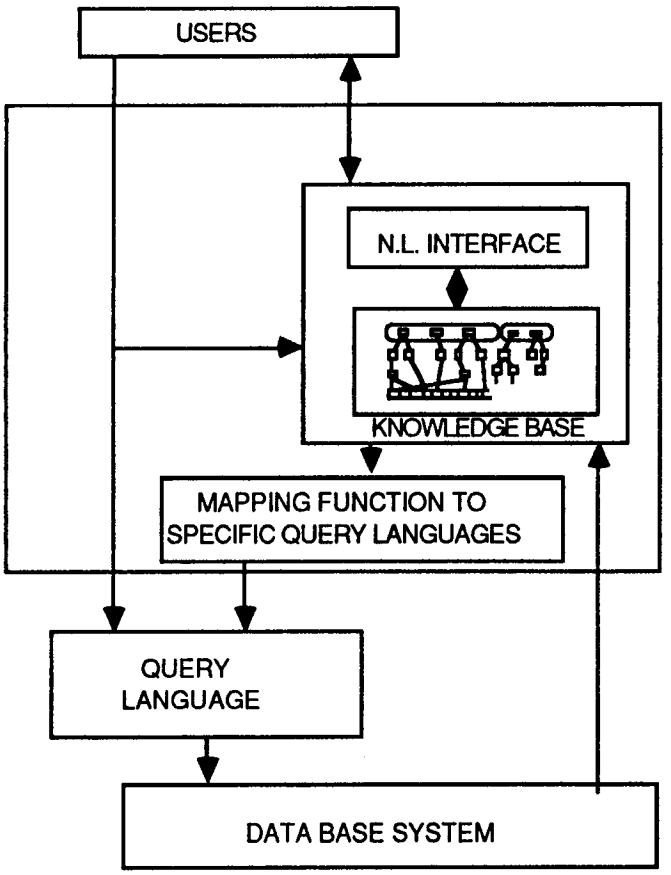
Before it can be used, the system needs to be trained. The difference between training and using the system is in the way humans communicate with it. To be sure that initially the system makes the correct connections between words and concepts, the system manager, or instructor, will give the system the appropriated information in the form of CD structures. These structures will be incorporated into the long term memory shaping up the initial dictionary and relations. At any time, if the system manager detects that the system is performing poorly, he can give the system additional information to reinforce proper CD structures and the connections between them. In this way, the correct links in the long term memory structure can be partially controlled.

Knowledge representation and knowledge acquisition.

The first issue when designing a knowledge base is to specify the formalism in which the part of the universe that is to be represented in the knowledge base is to be described. Conceptual Dependency (CD) structures are the basic formalism we have selected to represent information in our model. The key idea behind CD theory is that meaning can be represented in a canonical, language-free manner (Schank, 1972). In our implementation a CD is composed of an atomic head concept and a list of attribute-value (or role-filler) pairs where values are again CD structures. Complex events, concepts, or objects, can be represented by means of a network of CD structures, connected by different types of links, such as, causal, temporal or intentional (Pazzani, 1988).

For example, we can connect a CD structure representing the chemical element mercury as a heavy metal used on some industrial processes, with other CD structure representing the environmental problems caused by the uncontrolled disposal of heavy metals. In this manner, the user-expert module can hypothesize that a search of information about mercury, must include a search of documents about contamination caused by heavy metals.

In the knowledge base, general concepts (or patterns) are represented as a packed-CD structure (as opposed to a network of CD structures). These structures represent high order memory conceptualizations and



are generally known as Memory Organization Packets (MOP). MOPs are organized as a hierarchical and modifiable network that index individual cases or items of information (Schank, 1982; Kolodner 1983).

Other characteristic of our system is that most of the knowledge in it must be acquired using its own mechanisms and not pre-stored or hand-tailored. That means that the system considers all external information as cases (instances with no variables) and that the system itself must generate using its own learning mechanisms all the patterns that index those cases.

Inductive learning is the basic mechanism the system uses to create these new patterns. To learn new patterns, our prototype uses the empiricist algorithm making an input pattern more general or more specific, in response to feedback about how well that pattern explains or classifies an instance. To make a pattern more specific, our system indexes under the same pattern (MOP-father), all the instances or cases that are compatible with this MOP-father. A new MOP is created from these events when some of them share one or more features that are not present in the MOP father. This new MOP is indexed as a specialization of the MOP-father, and the events, used to create it, are indexed under it (Kolodner, 1983; Lebowitz, 1983).

To generalize its input, our program starts by replacing constants by variables (Fernandez-Valmayor & Fernandez Chamizo, 1992). SBL-vars (for Similarity Based Learning) are used to generalize input instances. These variables are created by the system when two CD structures have a common slot with values that are different, but with some possible common characteristics. For example:

The two CDs below are two possible representations of the chemical elements lead and mercury. (Following the usual convention we represent CD's head concepts in capital letters and attribute's names in non-capital letters; PP stands for picture producer following Shank's nomenclature and SBL-variables are written with names starting with char "¿").

```
(PP  type    INANIMATE-OBJECT
     is-a    CHEMICAL-ELEMENT    symbol   Hg
                                 group    METAL
                                 density  HIGH
     name    MERCURY)

(PP  type    INANIMATE-OBJECT
     is-a    CHEMICAL-ELEMENT    symbol   Pb
                                 group    METAL
                                 density  HIGH
     name    LEAD)
```

Using the generalization process described above our

system will obtain the following MOP-pattern:

```
(PP  type    INANIMATE-OBJECT
     is-a    ¿IS-A      symbol    ¿SYMBOL
                        group     METAL
                        density   HIGH
     name    ¿NAME)
```

In this pattern, the SBL-var ¿NAME can match any value in the slot 'name', but the SBL-var ¿IS-A only will match values in the slot 'is-a' with any head concept and attribute value pairs 'group-METAL' and 'density-HIGH'. Thus, the system can use this pattern to index information about all high density metals.

To create new patterns, our system uses another type of variables; EBL-vars (for Explanation Based Learning). These variables represent not merely a conjunctive generalization of several instances but the structural relation between the components of a unique instance. For example: The CD below describes a possible episode of water-contamination caused by the disposal of mercury waste

```
(ACTION    type        PTRANS
           actor       CHEMICAL-WASTE-LTD.
           object      MERCURY
           from        FACTORY
           located-in        FORESTLAND
                             manufacturer-of
                                  BATTERIES
                             owned-by
                                  CHEMICAL-WASTE-LTD.
           to     GROUND     located-in     ALMADEN
           date   1-5-89
           result      CONTAMINATION
                             type       WATER-
                                        CONTAMINATION
                             agent      MERCURY)
```

The EBL process will obtain the following pattern (in this pattern EBL-vars are marked with the symbol "?"):

```
(ACTION    type  PTRANS
           actor       ?ACTOR
           object      ?OBJECT
           from        FACTORY   located-in
                                 FORESTLAND
                                 manufacturer-of
                                 BATTERIES
                                 owned-by
                                 ?ACTOR
           to     GROUND     located-in     ALMADEN
           date   1-5-89
           result
           CONTAMINATION
                 type        WATER-CONTAMINATION
                 agent       ?OBJECT)
```

With this pattern the system captures the idea that the actor of the action, that result in water contamination, is the owner of the factory from which the object that causes contamination comes from. Specific domain knowledge is included in this process to deal with attributes (or roles) with special meaning in the domain.

Finally, our system has processes to change its 'focus of attention' or 'view point' about a given information.

In the examples above we have used to represent mercury the CD:

```
(PP  type    INANIMATE-OBJECT
     is-a    CHEMICAL-ELEMENT   symbol    Hg
                                group     METAL
                                density   HIGH
     name MERCURY)
```

Where the "focus of attention" is the CD-head concept 'PP'. Changing the "focus of attention" to 'MERCURY', the system obtains the following CD:

```
(MERCURY   name-of
     PP    type  NANIMATE-OBJECT
           is-a  CHEMICAL-ELEMENT   symbol   Hg
                                    group    METAL
                                    density  HIGH)
```

This CD represents the same information that the former but now, the focus is not a generic object (Picture Producer); the focus is MERCURY. This process of changing the "focus of attention" of a CD has three important consequences:

First, it makes possibly to match CDs that in other case will not be comparable; e.g., it makes possibly to match the CD above with the CD (MERCURY <empty list of attribute- value pairs>) that appear in the slots of the CD that describes the water-contamination case and as a consequence to inherit the attribute-value pairs of the former.

Second, these new 'view points' can be re-indexed in long term memory making new entries at the concept dictionary at the root of the memory. For example, changing the 'focus of attention' of the CD that describes mercury as a PP will create between others the following new entries:

```
(MERCURY   name-of     PP)

(Hg        symbol-of   CHEMICAL-ELEMENT)

(METAL     group-of    CHEMICAL-ELEMENT)
```

In this manner, the dictionary at the root of the memory will comprise all the terms used in the CDs given to the system.

Finally, when it is combined with the process of variabilization, changing the 'focus of attention' results in patterns that express different abstractions or

conceptualizations (although applied to the same information). For example, if in the water-contamination case we change the attention from ACTION to CONTAMINATION we obtain the following CD for the same information:

```
(CONTAMINATION
    result-of
    ACTION      type      PTRANS
                actor     CHEMICAL-WASTE-LTD.
                object    MERCURY
                from
                     FACTORY
                          located-in  FORESTLAND
                          manufacturer-of  BATTERIES
                          owned-by
                          CHEMICAL-WASTE-LTD.
                to   GROUND
                          located-in      ALMADEN
    date   1-5-89
    type   WATER-CONTAMINATION
    agent  MERCURY)
```

When EBL and SBL processes are applied to this CD the following pattern can be generated:

```
(CONTAMINATION  result-of
    ACTION      type      PTRANS
                          actor     ¿ACTOR
                          object    ?AGENT
                          from      ¿FROM
                               manufacturer-of
                                    BATTERIES
                          to   GROUND
                               located-in
                                    ALMADEN
          date      ¿DATE
          type      WATER-CONTAMINATION
          agent     ?AGENT)
```

This pattern expresses the abstraction that the contamination is caused by an agent that is the object translated from places where batteries are made to ground located in Almaden (Fernandez-Valmayor & Fernandez Chamizo, 1992)

User-expert module implementation

The NC prototype

The current implementation of the user-expert module is based on NC, a prototype written in Allegro Common Lisp. This prototype was first developed to test memory organization and learning algorithms (Fernandez-Valmayor, 1990). In NC, learning algorithms are used to create the abstractions that make the discrimination network that indexes all input information. Our system uses NC to index, by means of a network of patterns, the training cases and the cases that result of translating user's queries into CDs structures.

Our working hypothesis is that some basic but essential linguistic aspects of human memory can be modeled using two simple guide-lines:

- Memory is a schematic process, in which new information is integrated into already established structure of patterns (Barlett, 1932).
- There is a small set of processes responsible for maintaining this structure.

Memory structure and organization

From a structural perspective memory is divided into long term memory and short term memory. The long term memory is a sequence of trees, and the list of the roots of these trees constitutes its root. As the system incorporates new structures to long-term memory, this list is reorganized adding to it the CD head values that still are not in it. The process of changing the 'view point' also adds new entries to memory's root.

The nodes of the trees are the patterns that the system creates from input CD structures. The links that connect the trees' nodes have a label that expresses the difference between a child-node and its father and a frequency value; this frequency value is augmented each time the node is successfully accessed during a search operation.

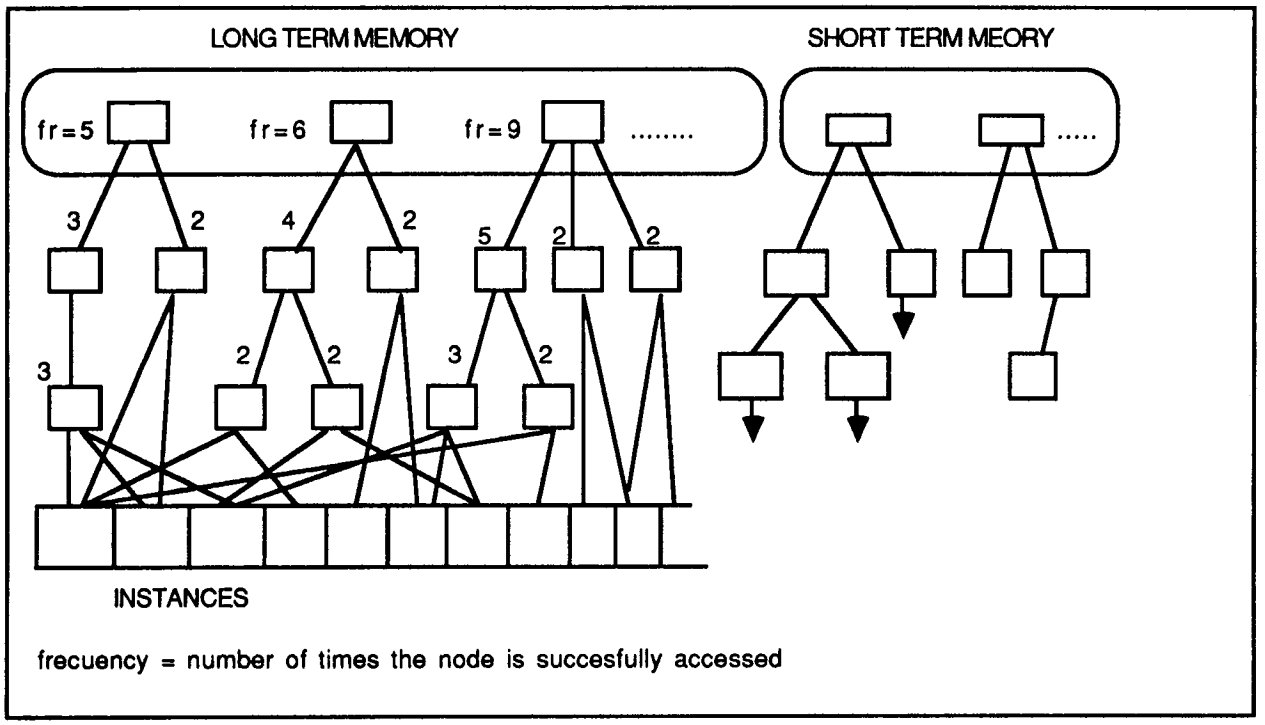
The leaves of the trees are the input CD structures, these CD structures can be the outcome of the translation of the user's input, or the CD structures by the system manager when training the system.

Short term memory is the sequence of CD structures that result from the translation of user input during the session. At the end of the session the short term memory is subsumed into the long term memory. Unlike the short term memory, which experiences rapid and shifting changes, the long term memory, though updated at the end of each session, is a more stable representation of the domain and user's peculiarities.

Memory based parsing.

The information retrieval problem is deeply related with the problem of natural language understanding. In the past years researchers in the IR field have been arguing that to build the next generation of information retrieval systems, we need systems able to extract the semantic information from both documents and user's queries, and that the matching of queries and documents must be done at this semantic level (Van Rijsbergen, 1987).

At the same time over the past years researchers in the NL area have developed a variety of techniques for natural language analysis (Gazdar & Melish, 1989; Schank & Riesbeck, 1981; Riesbeck & Schank, 1989). Underlying these techniques there are different approaches to NL processing: declarative versus procedural, or semantic versus pure syntactically driven parsing. However, all these approaches have in common that they are based on a search process plus a pattern matching analysis. What is different is the data structure they search and the kind of patterns they try to match. Thus in memory based parsing, understanding is considered as the process of searching memory for related language structures (Schank,



1980, 1982; Martin, 1989). Memory based parsing can be also considered as a continuation of the work initially developed in conceptual analyzers and script appliers.

In conceptual analysis, words are linked to conceptualizations in a dictionary. These conceptualizations are structures that represent the meaning of the word and they have associated procedural actions known as 'demons' or 'requests'. Conceptualizations not only have semantic meaning but also precise syntactic knowledge; for example in the conceptualization associated with the word "say"

```
(MTRANS   actor     X
          object    X
          from   X
          to     X)
```

we will have in the slot 'actor' the demon: "expect to have already hear of the human who is performing this act", and in the slot 'TO' the demon: "expect that the human receiving the message will be the object of the preposition 'to' " (Dyer, 1983).

In our approach, we try to use a more integrated and less procedural control structure than the requests-stack structure used in McELI (Birnbaum & Selfridge, 1981) or the working memory used in DYPAR (Dyer, 1983). In our system, the system's dictionary is the root of the long term memory, the list of trees' roots; and words in the input text are the primary cues to the information stored under them. The strength of associations between words and the information stored under its corresponding entry are different and from them the parser builds the short term memory as a graded list of expectations (alternative patterns) for the present input word. With this purpose, we define two functions, the 'weight' function and the 'activation' function.

The parsing algorithm

To develop a parsing algorithm based on our memory model we have to take into account three different but deeply related issues:

- How traditional parsing algorithms behave and what are the principles upon they operate.
- How to infer from examples the structure of the language.
- How to deal with the inherent ambiguity of natural language.

First issue. It is claimed (Tomita, 1986) that the syntax of natural language can be realized as a context free grammar. Moreover, semantic grammars are also context-free phrase structure grammars in which semantics and syntaxis is encoded in form of productions. Many general context-free parsing algorithms, able to handle arbitrary context-free grammars, have been formulated. Among them, the most well-known is Earley's algorithm. All practical algorithms have some similarity with Earley's algorithm. General parsing algorithms are like Earley's algorithm, in that they all construct well-formed sub-string tables (Tomita, 1986). At any point in its execution the parser can avoid to make redundant analysis by looking for phrases already recognized in its table of well-formed sub-strings. When all the phrases of a given type are stored in the

table there is no longer need to use the grammar.

We conceive our long term memory as a discrimination network of patterns equivalent to a table of well-formed sub-strings. These patterns play a role comparable to that of the "items' list" that constitute the table in Earley's algorithm (Aho &Ullman, 1977). In our case, 'items' are the patterns indexed under each root's memory entry, and the long term memory is used by the parsing algorithm in a way similar to that of 'traditional' parsing algorithms, that is we use them to alternate top-down prediction with bottom-up production reduction (Pereira & Shieber, 1987).

Second issue. The problem of learning a language from a training set of sample sentences is the problem of grammatical inference (Angluin & Smith, 1983). In our system initial learning happens when the instructor presents the system an initial set of CD structures. A CD is a positive instance to be learned. A CD is a tree structure (with labeled branches) that we consider as equivalent to the derivation tree of the input sentence. Thus, the system must learn the language from structural presentation, a technique used by pattern recognition researchers to aid grammatical inference. As we have seen above, parsing algorithms only need to use the grammar to build the well-formed sub-string table, so our problem is not to infer the grammar but to built the equivalent of the well-formed sub-string table. That is what the implemented learning mechanisms do.

Third issue. Usually natural language grammars are ambiguous; a sentence can have multiple parses. In the case where there exists two or more possible parses, it is not acceptable for a practical natural language parser to produce only one arbitrary parse. All possible parses should be produced and stored somewhere for later disambiguation. When inferring a grammar we have another source of ambiguity: the same language may be generated by several different grammars. Thus, the question 'which grammar will the system learn?' arise.

As we have seen what our system does is to build the equivalent of the parsing list of items needed to parse the input sentence. So if there are different underlying grammars in the training CDs instances the well-formed sub-string table will correspond with more than one grammar. Therefore, when searching the memory it will be founded expectations that correspond not only to different parses in the same grammar but to parses in different grammars.

What follows is an outline of how the parsing of an input proceeds:

1- System initializes short term memory

2- It searches for the first word in the input in the operational memory. It makes active the patterns under it by putting them in the short term memory.

3- The activated patterns generate expectations about words that would be seen next in the input. These expectations will be tested against the patterns activated by the following word input. Active patterns in short term memory not compatibles with following words in the input are discharged.

4- If it doesn't find the word, it leaves a hole for that word in the short term memory and proceeds with next input word.

Conclusions and future work.

In this paper we have seen that a human memory model is not a good candidate to replace actual database models. Its potential lies in its merit to model a human expert that helps the user to formulate better queries.

Natural language processing capabilities are at the core of the system described here. The focus of our paper has been to develop further memory based parsing techniques. If we compare the techniques we use in our system with previous work, we can say that what we have made is to substitute the static dictionary (which guides the parsing) and much of its hand-tailored procedural knowledge by a dynamic structure and a set of generic processes.

Learning patterns of well-formed sub-strings from positive examples has been the other focal point of our paper. The representation scheme and learning mechanisms described here can be considered close to those of Winston's learning blocks world concepts, as they are described in Michalski, Carbonell & Michell (1983). The CD network used in our system and the semantic network used by Winston are similar, but in our system the network of concepts and relations is more homogeneous and it is processed in a less hand-crafted way. The general process that in our system changes the 'point of view' over a CD can be considered related to the negative-satellite links used by Wiston, but in our case this is not a particular information introduced by the user, but a general process that expands the learning power of the system.

Other question that has been arisen is the need to compare the representational power of CDs structures with logic formulas. External links between CDs and slot names can be thought as two-argument predicates but in our system we have also incorporated logic operators ('or', 'and', 'not') as links that are processed in a special manner by functions like the function that changes the focus of attention.

Textual analysis of accurately retrieved information can be the next goal to improve the context in which user questions and comments must be interpreted. Evaluation of text-analysis technologies has demonstrated that text-analysis techniques, incorporating natural language-processing, are more effective than traditional information-retrieval techniques based on statistical classification when applications require structured representations of the information present in texts (Lehnert & Sundheim, 1991)

References

Aho, A.V. & Ullman, J.D. (1977). *Principles of Compiler Design*. Addison Wesley.

Angluin, D. & Smith, C.H. (1983). *Inductive Inference: Theory and Methods, Computing Surveys*, Vol. 15,No.3, pp. 237-269.

Barlett, F.C., (1932). *Remembering. A Study in Experimental and Social Psychology*. Cambridge University Press. (2a. reimpresión, 1977).

Birnbaum, L. & Selfridge, M.,(1981). Conceptual analysis of natural language. En Schank, R.C. & Riesbeck C.K. ,(Ed.),1981. *Inside Computer Understanding*, Lawrence Erlbaum Associates, Inc., New Jersey.

Borgman, C.L., Belkin, N.J., Croft, W.B., Lesk, M.E., Landauer, T.K. (1988). Retrieval systems for the information seeker: Can the role of the intermediary be automated? *Panel Session Human Factors in Computing Systems*, CHI' 88 Conference Proceedings (Washington, D.C.), ACM, New York, May 1988, 51-53.

Blair, D.C., & Maron, M.E. (1985). An evaluation of retrieval effectiveness for a full-text document retrieval system. *Communications of ACM*, 28, 289-299.

Dyer, M.,(1983). *In Depth Understanding*. MIT Press.

Fernández-Valmayor, A. (1990). Diseño de una Base de Conocimientos Dinámica y su Aplicación en un Entorno Educativo. Doctoral thesis. Universidad Complutense Madrid.

Fernández-Valmayor, A. & Fernández Chamizo, C. (1992). Educational and Research Utilization of a Dynamic Knowledge Base. *Computers & Education*, Vol.18, No. 1-3, pp. 51-61.

Fernández-Valmayor, A. & Villarrubia, C. (1991). Propuesta de Sistema Unificado de Consulta de Bases de Datos documentales para el Instituto de la Mediana y Pequeña Empresa Industrial, IMPI. Inform Técnico. Departamento de Informática, Universidad de Castilla-La Mancha, Ciudad Real.

Foltz, P.W. (1991). Models of Human Memory and Computer Information Retrieval: Similar Approaches to Similar Problems. ICS Tech Report 91-03. University of Colorado.

Gazdar, G. & Mellish, C., (1989). *Natural Language Processing in Lisp*. Addison-Wesley.

Kolodner J. L.,(1983). Maintaining Organization in a Dynamic Long-Term Memory, *Cognitive Science*, Vol.7, pp.243-280.

Kolodner J.L., (1983). "Reconstructive Memory: A Computer Model". *Cognitive Science*, Vol.7, pp.281-328.

Lehnert, W. & Sundheim, B. (1991). A Performance Evaluation of Text-Analysis Technologies. *AI Magazine*, Fall 1991, pp. 81-94.

Lebowitz, M., (1983). Generalization from Natural Language Text. Cognitive Science vol 7, pp.1-40.

Michalsky, R.S., Carbonell, J.G., Mitchell, T.M. (1983). *Machine Learning: an Artificial Intelligence Approach*. Morgan Kaufmann.

Martin,C.E.(1989). Case-based Parsing in Riesbeck, C.K. & Schank, R.C. *Inside Case-Based Reasoning*. Hillsdale, N.J. Lawrence Erlbaum Associates.

Pazzani, M.,(1988). Learning Causal Relationships: An Integration of Empirical and Explanation-Based learning Methods. PhD. Dissertation, *Technical Report UCLA-AI*-88-10.

Pereira F.C.N. & Shieber S.M. (1987). *Prolog and Natural-Language Analysis*. CSLI lecture notes; no.10

Riesbeck, C.K., (1986). From Conceptual analyzer to Direct Memory Access Parsing: An Overview. In *Advances in Cognitive Science*, Sharkey N.E. (Ed.). John Wiley & Sons.

Riesbeck, C.K. & Schank, R.C., (1989). *Inside Case-Based Reasoning*. Hillsdale, N.J. Lawrence Erlbaum Associates.

Salton, G. (1986) Another look at automatic text-retrieval systems. *Communications of ACM*, Vol 29, No.7, 648-656.

Salton, G. (1989) *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley.

Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill: New York.

Schank, R.C., (1982). *Dynamic Memory: A theory of reminding and learning in computers and people*. Cambridge University Press.

Schank, R.C., (1980). *Language and Memory*. Cognitive Science, 4, 243-284

Schank, R.C. , (1972). "Conceptual Dependency: A Theory of Natural Language Understanding", *Cognitive Psychology*. 1972, 552-631.

Schank, R.C. & Riesbeck C.K. . Ed. (1981). *Inside Computer Understanding*. Lawrence Erlbaum Associates, Inc., New Jersey.

Tomita, M. (1986). *Efficient Parsing for Natural Language: A Fast algorithm for Practical Systems*. Kluwer Academic Publishers

Van Rijsbergen, C.J. (1987). A new theoretical framework for information retrieval. *SIGIR FORUM* Vol. 21, No 1-2 Autum 86/Winter 87