An Experiment in Intelligent Information Systems: RESEARCHER

Michael Lebowitz
Department of Computer Science
Columbia University
New York, NY 10027
November, 1985

CUCS-171-85

To appear in *Intelligent Library and Information Systems*,
edited by R. Davies, published by Ellis Horwood.

An Experiment in Intelligent Information Systems: RESEARCHER

Michael Lebowitz[1]
Department of Computer Science
Computer Science Building, Columbia University
New York, NY 10027, USA

15 November 1985

Abstract

The development of very powerful intelligent information systems will require the use of many Artificial Intelligence techniques including some derived by studying human understanding methods. RESEARCHER is a prototype intelligent information system that reads, remembers, generalizes from and answers questions about complex technical texts, patent abstracts in particular. In this paper, we discuss three areas of current research involving RESEARCHER -- the generalization of hierarchically structured representations; the use of long-term memory in text processing, specifically in resolving ambiguity; and the tailoring of answers to questions to the level of expertise of different users. All of these areas are crucial for truly powerful information systems. We outline our methods and give examples of RESEARCHER processing various examples.

1 Introduction

Traditional work in information retrieval (such as that described in [Heaps 78; Salton and McGill 83]) has concentrated on ways of storing and retrieving texts based on their lexical contents with little regard to meaning. While this has led to quite powerful and useful systems, we can hope to do still better by applying techniques of Artificial Intelligence to information retrieval. Specifically, we will describe in this paper how we are using research in the areas of natural language processing and learning to help understand how to develop powerful, intelligent information systems. General discussion of the application of Artificial Intelligence to information retrieval can be found in [Schank et al. 80; DeJong 83; Lebowitz 83a].

---

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165. Many people have contributed to RESEARCHER. In particular, the work on generalizing hierarchies has largely been conducted by Kenneth Wasserman and the work on question answering by Cecile Paris in conjunction with Kathleen McKeown. Comments on an earlier draft of this paper by J. A. Campbell and Roy Davies were most useful.

In [Lebowitz 83a; Lebowitz 83b], we described the early stages of development of RESEARCHER, a prototype intelligent information system. RESEARCHER is intended to accept information in natural language form, in particular, patent abstracts such as EX1 .

EX1 - P137; U. S. patent #4400750; Forestlane Co. Ltd.

A magnetic read/write head carriage assembly for a floppy disk drive is disclosed for use with double sided floppy disks which permits the head tracking force to be easily and accurately adjusted. The head carriage assembly comprises a coil spring, having a central coil portion and first and second ends, which is mounted in a position between the base and the head support arm of the carriage assembly with the first end coupled to the base and the second end coupled to the support arm. An adjusting screw is mounted on the base adjacent to the first end of the coil spring for adjusting the position of this end, thereby adjusting the biasing force applied by the spring to the support arm.

RESEARCHER:

- understands the text (extracts the meaning).
- adds the acquired information to a long-term memory, learning through generalization as it does so.
- answers questions about the information in its memory.

Descriptions of complex physical objects such as those in patent abstracts leads to special problems in the development of intelligent information systems. In this paper, we will present an overview of three areas that we are studying using RESEARCHER that are important in the development of intelligent information systems:

- learning by generalizing hierarchical descriptions -- learning is important if we wish our systems to be able to return *more* information than we give them, by comparing the different texts read; physical objects can be best represented hierarchically.
- using a long-term memory built up through generalization to assist in text interpretation -- intelligent information retrieval requires robust understanding, and this can only be done with extensive application of knowledge, including that which the system acquires.
- tailoring the system's answers for different users -- a truly intelligent system with a large memory can be most effective by giving answers that are crafted for each individual user.

We begin by describing the problems of generalizing hierarchically structured objects, as the creation of a long-term memory is necessary for our text understanding methods.

2 Generalizing Hierarchical Descriptions

Intelligent information systems should be able to return to a user more information than that contained in any single text. They should be able to *learn* from the texts by noticing similarities (among other learning methods). The patent abstracts that we have been looking at describe the physical structure of complex objects. Since such objects are naturally represented as hierarchies of parts, our learning research has addressed the generalization of hierarchically structured descriptions. To see the input to the generalization process, consider EX2, the first part of a typical patent abstract.

> **EX2** - P37; U. S. patent #4190870; Avina Raymond, Merrell Patrick
>
> A disk drive assembly includes a baseplate housing joined with front and rear covers to enclose a spindle that supports ball bearings and a hub for rotating a stack of magnetic disks ...

Our representation of patent abstracts such as this one includes three classes of information:

- *a parts hierarchy*, that indicates the components of each part.
- *interpart relations*, capturing physical and functional relations between various components.
- *properties* of the objects.

The level of detail needed for each of these classes of information certainly depends on the task at hand. For our purposes, we have concentrated on the parts hierarchy and physical relations (which are represented using a primitive-based canonical scheme [Wasserman and Lebowitz 83]). We are currently working on classification schemes for functional relations and object properties (such as size and composition), which are crucial in understanding many device descriptions. Figure 1 shows the representation of EX2 created by RESEARCHER.

We can see in Figure 1 all three of the types of information used in representing physical description patents. The backbone of the representation is a parts hierarchy. (The numbers in the hierarchy refer to the objects listed on the right.) Figure 1 shows an assembly (part 1) composed of a number of parts, including a disc drive, an enclosure, and so forth. One of the components is another assembly (the "stack") which is composed of a number of discs. There is also a "loose part", the baseplate, not

```
Text Representation:

              |-------------2                  1 = UNKNOWN-ASSEMBLY# ('ASSEMBLY')
              |----------AB-3                  2 = DISC-DRIVE#
              |-----------B-5                  3 = ENCLOSURE#
              |-----------D-6                  5 = NUMBER/>1 LOCATION/FRONT,REAR COVER#
----------C-1|----------CD-7                  6 = DRIVE-SHAFT#
              |-----------E-8                  7 = NUMBER/>1 BALL-BEARING#
              |-----------E-9|------------10    8 = HUB#
                                               9 = UNKNOWN-ASSEMBLY# ('STACK')
                                              10 = NUMBER/>1 DEV-TYPE/MAGNETIC DISC#

----------------------A-4   4 = BASEPLATE#

A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM4 (BASEPLATE#)    (UNKNOWN-PURP-REL) &MEM3 (ENCLOSURE#)
[&REL2/B]  &MEM3 (ENCLOSURE#)    (R-CONNECTED-TO)   &MEM5 (COVER#)
[&REL3/C]  &MEM6 (DRIVE-SHAFT#)  (R-SURROUNDED-BY)  &MEM1 ('ASSEMBLY')
[&REL4/D]  &MEM6 (DRIVE-SHAFT#)  (P-SUPPORTS)       &MEM7 (BALL-BEARING#)
[&REL5/E]  &MEM8 (HUB#)          (P-ROTATES)        &MEM9 ('STACK')
```

Figure 1: A typical RESEARCHER representation

included in the main assembly. The items in Figure 1 with #'s indicate concepts (e.g., disc-drive#, the concept of a disc drive) as opposed to words (the phrase, "disk drive").

Figure 1 also includes several relations between objects. These are shown by letters in the parts hierarchy and enumerated below the hierarchy. For example, the "B" with the enclosure (part 3, from the word "housing" in EX2) and the covers (part 5) indicates that they are connected ("joined") to each other.

Finally, the representation of EX2 includes several object descriptions augmented with properties. These are indicated in Figure 1 in the object descriptions. For example, object 10, the discs, has been modified by making its device-type property "magnetic". The object representations of the covers, ball bearings and disc also include indications that there is more than one of each.

The representation in Figure 1 is at a level of understanding that many Artificial Intelligence text processing systems might achieve if applied to this domain. However, full understanding requires that we integrate this representation with existing knowledge in memory. In particular, RESEARCHER has as one of its goals the incremental generalization of descriptions such as that in Figure 1 by finding similar examples in memory, comparing the new example with them, and abstracting out any similarities. Since

there has been considerable work done on generalizing objects described with property values and to some extent relations ([Winston 72; Michalski 80; Lebowitz 83c], among others), we have concentrated on the problems of generalizing hierarchically structured objects.

Generalizing hierarchical representations presents a number of difficult problems. Typical are the problem of deciding how the components in the objects being compared correspond; dealing with differing levels of description of objects; and structuring memory so that maximal inheritance of the sort used in semantic networks and frame systems (see [Barr et al. 82]), which implies minimum space utilization, can be achieved automatically. In this paper, we will give examples of how the generalization process works, and refer the reader to [Wasserman 84; Wasserman 85] for more details.

We can break generalization into two parts: 1) since RESEARCHER is not explicitly being taught concepts, when a new example is presented it must decide what other objects to compare it to, and 2) the process of comparing object descriptions, which includes the abstraction of similarities.

We will look at the comparison process first, as it is involved in the search process. Figure 2 shows two simplified disc drive patents.[2] The physical relations and properties involved are not displayed, though they are handled in the generalization process.

As human understanders, we can easily see that patents EX3 and EX4 describe similar objects, particularly after looking at the hierarchical representations. However, to begin to generalize the similarities, RESEARCHER must decide how the parts of the representations correspond -- for example, that part 2, the enclosure in EX3, corresponds to part 11, the similar assembly in EX4, and not to parts 12, 13 or 14. Here this is relatively easy, as the assemblies are identical, but we must be able to identify less than perfect matches. RESEARCHER does this with a numerical scoring algorithm, similar to the one in [Winston 80] and related to [Evans 68].

---

[2]Simplified versions are used as the complex nature of real abstracts obscures the generalization process as well as to avoid text processing problems.

```
Patent: EX3

(A DISC DRIVE COMPRISING AN ENCLOSURE SURROUNDING THE DISC DRIVE *COMMA*
SAID DISC DRIVE INCLUDES A SPINNING ASSEMBLY A DISC AND A READWRITE HEAD
*COMMA* SAID SPINNING ASSEMBLY INCLUDES A SPINDLE CONNECTED TO A MOTOR
*COMMA* SAID ENCLOSURE COMPRISING A COVER ON TOP OF A SUPPORT MEMBER)

Text Representation:

              |              |-------------8    1 = DISC-DRIVE#
              |-------------2|-------------9    2 = ENCLOSURE#
              |                                 3 = DEV-TYPE/ROTATION
              |                                     UNKNOWN-ASSEMBLY# ('ASSEMBLY')
-------------1|              |-------------6    4 = DISC#
              |-------------3|-------------7    5 = DEV-TYPE/READ/WRITE HEAD#
              |                                 6 = DRIVE-SHAFT#
              |-------------4                   7 = MOTOR#
              |-------------5                   8 = COVER#
                                                9 = UNKNOWN-THING# ('MEMBER')

Patent: EX4

(A DISC DRIVE COMPRISING AN ENCLOSURE SURROUNDING THE DISC DRIVE *COMMA*
SAID DISC DRIVE INCLUDES A SPINNING ASSEMBLY A MAGNETIC ASSEMBLY AND A
READWRITE HEAD *COMMA* SAID SPINNING ASSEMBLY INCLUDES A SPINDLE CONNECTED
TO A MOTOR *COMMA* SAID MAGNETIC ASSEMBLY COMPRISING A DISC *COMMA* SAID
ENCLOSURE COMPRISING A COVER ON TOP OF A SUPPORT MEMBER)

Text Representation:

              |              |------------18   10 = DISC-DRIVE#
              |------------11|------------19   11 = ENCLOSURE#
              |              |                 12 = DEV-TYPE/ROTATION
              |                                    UNKNOWN-ASSEMBLY# ('ASSEMBLY')
------------10|              |------------15   13 = DEV-TYPE/MAGNETIC
              |------------12|------------16       UNKNOWN-ASSEMBLY# ('ASSEMBLY')
              |                                 14 = DEV-TYPE/READ/WRITE HEAD#
              |------------13|------------17   15 = DRIVE-SHAFT#
              |------------14                   16 = MOTOR#
                                                17 = DISC#
                                                18 = COVER#
                                                19 = UNKNOWN-THING# ('MEMBER')
```

Figure 2: Two simple device representations

Figure 3 shows the output of the generalization process for these objects, taken from [Wasserman 85]. It is assumed that EX3 is already in memory and EX4 is being compared to it. We can see how RESEARCHER first makes correspondences of the sort mentioned above. One problem arises in dealing with the discs (parts 4 and 17). The disc in EX3 is described as part of the disc drive, while in EX4 the disc is part of a magnetic assembly which is part of the disc drive. To make the representations match, RESEARCHER must insert a "null" part, which may or may not actually exist in any given object. This is legitimate as we can assume that the descriptions are incomplete or use different levels of aggregation. The null part appears in the generalization of EX3 and EX4as part 24, shown at the bottom of Figure 3.

The two input representations are stored as variants of the generalized object, recording only how they differ from it (basically, in this case, how the null object is resolved; in real examples there would usually be more differences).

```
(GEN '&MEM10)
Matching &MEM10 against &MEM1 .... 170
Best match is:
(170 ((&MEM1 . &MEM10)
      ((&MEM5 . &MEM14))
      ((NULL# . &MEM13) ((&MEM4 . &MEM17)))
      ((&MEM3 . &MEM12) ((&MEM7 . &MEM16)) ((&MEM6 . &MEM15)))
      ((&MEM2 . &MEM11) ((&MEM9 . &MEM19)) ((&MEM8 . &MEM18)))))
Incorporating into g-tree ...
New generalization created: &MEM22
with variants: (&MEM10 &MEM1)

********************

              |------------23                 22 = DISC-DRIVE#
              |------------24|------------25  23 = HEAD#
              |                              24 = NULL#
              |              |------------27  25 = DISC#
------------22|------------26|------------28  26 = UNKNOWN-ASSEMBLY#
              |                              27 = MOTOR#
              |------------29|------------30  28 = DRIVE-SHAFT#
                             |------------31  29 = ENCLOSURE#
                                            30 = UNKNOWN-THING#
                                            31 = COVER#
```

Figure 3: Generalizing the representations in Figure 2

Even with just two hierarchical descriptions to compare, the matching process involves a number of problems in determining how the components of the hierarchies correspond. Typical of such problems is the need to insert levels in a hierarchy to obtain a good match, as described above. (While the insertion of a null level by itself decreases the goodness of a match, it will often greatly increase the value of lower level matches.) The problem is that there are an exponentially large number of places where null levels can be inserted, each requiring a complex, recursive match to test. We have used RESEARCHER to experiment with a variety of different algorithms for deciding where null levels should be inserted for optimal matching, concentrating on ones that only try the most obvious places near the top of the hierarchy.

Since the examples given RESEARCHER are not expressly designed for learning specific concepts (as they would be for a system being taught concepts, e.g., Winston's arch program [Winston 72]), the program must decide which examples to compare for the purpose of generalization. This is done using a

generalization-based memory of the sort in [Lebowitz 83c; Lebowitz 83d]. A hierarchy of concepts that organizes specific examples is created in memory. A possible memory is shown in Figure 4, where there are two concepts subordinate to disc-drive#, floppy and hard disc drives. The former has two further sub-concepts. Each concept organizes a group of instances. Note that: 1) using the techniques outlines in this section, the generalization hierarchy is automatically created by RESEARCHER, not provided to the system in advance (at least this will be the case in a fully developed system; see [Lebowitz 83d; Lebowitz 84; Wasserman 85] for progess to-date); 2) each node in Figure 4 represents a complete hierarchical description of the kind we have been looking for (in effect, RESEARCHER's memory is a hierarchy of hierarchies); and 3) information in the generalization notes can be inherited by lower level generalizations and examples, so that information need not be stored repetitively.

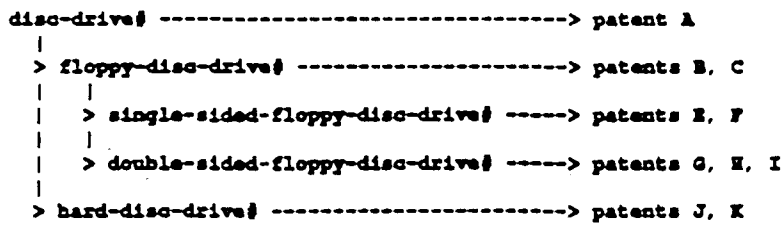

Figure 4: A typical generalization-based memory

In using its generalization-based memory, RESEARCHER takes each new example and searches down the tree for the example or generalized concept most similar to it. This process involves matching generalized concepts with the new example in much the same way as EX3 and EX4 were matched above. RESEARCHER begins by matching the new example with each of the children of the generalization tree's root. It then selects the best match and looks at that node's children. As long as one of the children produces a better match than the parent node, RESEARCHER continues down the tree. Eventually, it either reaches a leaf (an instance already in memory) or a maximally good generalization (i.e., all of the subordinate nodes contain factors that decrease the quality of the match).

RESEARCHER's search algorithm does not guarantee finding the best match, since it is possible that only an inferior match at an interior node would lead to the optimal match. However, this is viewed as

an acceptable compromise in limiting search. An alternative would be to set a threshold match value and search down all branches of the tree that match above this threshold. Again, though, we might still miss the best match. Only experimentation will tell whether one algorithm or the other improves chances of finding a good (if not always best) match.

Sacrificing a guarantee of finding the maximally similar example (in return for computational feasibility) matches the observation that human memory gains robustness by initial heuristic classification of information. (E.g., if we first classify a person we meet as an Artificial Intelligence researcher, we may miss similarities he or she has with our favorite politician.)

Eventually, after a series of matches, RESEARCHER selects the node in memory which it believes the new example best matches, either a previous example or an existing generalization. It then "factors out" similarities between these representations, and, if need be, creates a new generalization node. In any case, the new example is stored by recording how it differs from a generalization in memory. This is an optimally space-efficient method of storage, which also captures significant generalizations about the objects in the domain.

The current implementation of RESEARCHER's generalization scheme works quite well on modest-sized examples. In addition to disc drive patents, a modified version of the program (CORPORATE-RESEARCHER [Wasserman 85]) has been tested on hierarchical descriptions of corporate organizations. In the future, we plan to address some of the combinatoric problems that arise for large examples, consider whether our approach of abstracting out all possible similarities is too extreme (in particular, determining exactly what the generalized concepts signify) and applying confidence evaluation methods of the sort described in [Lebowitz 82] to refine initial generalizations.

3 Text Processing Using Memory

Since intelligent information systems such as RESEARCHER have available many examples in memory, it seems natural to make use of this information for text processing (beyond identifying lexical items [Harris 78]). If we wish to have intelligent systems with large amounts of information it is vital that text processing be robust. Patent abstracts, like the rest of natural language, are quite ambiguous (which was somewhat surprising to those of us a bit naive about patents, who expected perfect clarity). We can use the system's automatically updated memory to help resolve many ambiguities.

We feel that the best way to use detailed memory information during text understanding in the context of current systems is to identify specific tasks where a piece of information from memory will be useful. More general methods, such as using memory to determine the interesting aspects of a text to focus processing [Schank 79; Lebowitz 81], we leave for the future. We have identified a set of "questions" that arise during text processing that can most easily be answered (and often can *only* be answered) by accessing long-term memory.

It is important to keep in mind that we are proposing using *memory* for understanding, as opposed to general semantic information about words or concepts (as many other Artificial Intelligence systems have done). While general information is crucial for our conceptually-based understanding methods, in order to resolve many ambiguities, it will be necessary to look at very detailed information in memory -- in our case, how the objects described in patent abstracts are constructed and how their pieces relate to each other. One way of looking for at this distinction is that RESEARCHER ends up using similar information to other Artificial Intelligence but much less of it has to be hand-coded initially.

EX5 illustrates two kinds of ambiguities that arise in patent abstracts.

EX5 - A disc head supporting a spindle made of magnetic material.

The first ambiguity in EX5 involves "disc head". Although not syntactically ambiguous, an understanding system such as RESEARCHER must determine the conceptual relationship between the

nouns. The phrase "made of magnetic material" is ambiguous in that we do not know whether it attaches to the head or the spindle. Both of these ambiguities can only be resolved by looking at memory. In fact, it would be easy to construct scenarios where different states of memory would cause this example to be understood differently (e.g., whether we knew about magnetic heads or magnetic spindles).

RESEARCHER makes use of relatively simple, but heavily memory-based, techniques for handling ambiguities of the sort in EX5. Its conceptual analysis type text processing algorithm (described in [Lebowitz 83b; Lebowitz 84]), involves identifying object descriptions (usually noun groups) and connecting them with various relational words (usual prepositions -- patent abstracts are quite short of verbs) which indicate the various physical, functional and assembly / component relations mentioned in Section 2. Within this processing algorithm, we have identified places where ambiguity can be identified and memory queried for resolution. In particular, memory is asked which of several possible physical constructions is more likely or what relation is likely to occur between two objects. Questions in both classes are answered by looking for examples of the possible configurations that already exist in memory.

Figure 5 lists the various questions that RESEARCHER can currently ask its memory for purposes of disambiguation. They primarily involve prepositional phrase attachment and noun groups with multiple nouns.[3] Our analyses of these ambiguities shares much with the linguistic work of [Levi 78] and the application of this work to Artificial Intelligence in [Finin 82]. However, our method of resolving the ambiguities -- the use of a dynamic, long-term memory -- is rather different. By looking for specific examples in memory (or information generalized from specific examples) we allow RESEARCHER to always use the best information currently available. While examples do not always provide resolution of ambiguities as clearly as pre-defined semantic properties, they have the clear advantage of minimizing the need for *ad hoc*, hand-coded information.

---

[3]The word types used in Figure 5 are functional, rather than syntactic. However, object words are usually nouns and relation words and part indicators are usually prepositions, although not always in either case.

Form: object-word1 object-word2
Example: An actuator housing ...
Question: What is the relation between object-word1 (actuator) and object-word2 (housing)?

Form: modifier object-word1 object-word2
Example: A metal drive cover ...
Question: Does the modifier (metal) better apply to object-word1 (drive) or object-word2 (cover)?

Form: object-word1 object-word2 object-word3
Example: A disc-drive transducer wire ...
Question: Is object-word1 (disc-drive) "related to" (as a part, assembly or in relation) object-word2 (transducer) or object-word3 (wire)?

Form: object-word1 relation-word1 object-word2 relation-word2 object-word3
Example: A transducer on top of a disc supported by a spindle ...
Question: Does relation-word2 (supported by) connect object-word3 (spindle) with object-word1 (transducer) or object-word2 (disc)?

Form: object-word1 part-indicator1 object-word2 part-indicator2 object-word3
Example: A disc drive including a disc comprising a metal plate (and) ...
Question: Is object-word3 (metal plate) a part of object-word1 (disc drive) or object-word2 (disc)?

Form: object-word1 relation-word object-word2 part-of-indicator object-word3 (There are several related configurations.)
Example: A disc on a spindle for a disc drive ...
Question: Is object-word1 (disc) or object-word2 (spindle) a part of object-word3 (disc drive)? (Directly a part, as both are parts indirectly.)

Figure 5: RESEARCHER disambiguation questions

The search for possible examples that answer a given question is a relatively simple one. Recall that RESEARCHER's memory (Figure 4) is basically a hierarchy of object descriptions. In addition to the hierarchy for the main concept under consideration (e.g., disc drives), there are subsidiary hierarchies for other objects, such as spindles and discs. RESEARCHER uses these hierarchies to look for possible assembly/component constructions and physical relations. It begins its search with general object descriptions and searches through more specific descriptions until a relevant example is found. If several possible constructions (or relations) are found, the one associated with the most general description is used, as that represents RESEARCHER's broadest information. RESEARCHER's memory search disambiguation process is described in more detail in [Lebowitz 84].

Our disambiguation methodology bears resemblance to that of [Small 80; Birnbaum and Selfridge 81; Hirst 83], except, crucially, it relies on information from a detailed, dynamic memory for executing disambiguation. Our algorithm does have the side-effect of making understanding subjective (in the sense of [Abelson 73; Carbonell 81]), since new examples will be interpreted to correspond to old ones, but we view this as inevitable if we wish to achieve robust understanding.

To illustrate RESEARCHER's memory-based disambiguation, we will first consider how the simple noun phrase, "a motor spindle", is processed. Figure 6 shows how RESEARCHER processes this phrase with no relevant information in memory.[4] The program queries memory (indicated by ">>>" for a plausible relation between a motor and a drive shaft (spindle). Since none is found, it assumes that there is an unknown functional (purpose) relation between the objects, which is its default for *concrete* objects.

```
Running RESEARCHER at 3:19:26 PM

(A MOTOR SPINDLE)

Processing:

A               : New instance word -- skip
MOTOR           : Memette within NP; save and skip
SPINDLE         : MP word -- memette DRIVE-SHAFT#
  New DRIVE-SHAFT# instance (&MEM0)
  >>> Looking for relation between MOTOR# and &MEM0 (DRIVE-SHAFT#)
  New MOTOR# instance (&MEM1)
  Assuming &MEM1 (MOTOR#) and &MEM0 (DRIVE-SHAFT#) are functionally related
  Establishing UNKNOWN-PURP-REL; SUBJECT: &MEM1 (MOTOR#);
    OBJECT: &MEM0 (DRIVE-SHAFT#) [&REL1]

Text Representation:

----------------------A-0    0 = DRIVE-SHAFT#

----------------------A-1    1 = MOTOR#

A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM1 (MOTOR#)        (UNKNOWN-PURP-REL) &MEM0 (DRIVE-SHAFT#)
```

Figure 6: 'A motor spindle' with memory empty

Now we assume that before processing "a motor spindle", RESEARCHER had had in memory EX6,

[4] In Figure 6 and other examples of RESEARCHER output, the term "memette" refers to an object concept in memory – literally a small piece of memory. An MP, or Memory Pointer, is a word that points to an object in memory. Normally MPs are concrete nouns.

shown in Figure 7 along with its representation. EX6 simply consists of a drive with two parts, a motor on top of a drive shaft.

```
EX6 - A drive with a motor on top of a drive shaft.

Text Representation:

                  |---------------A-1   0 = DRIVE#
-----------------0|---------------A-2   1 = MOTOR#
                                        2 = DRIVE-SHAFT#

A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM1 (MOTOR#)        (R-ON-TOP-OF)      &MEM2 (DRIVE-SHAFT#)
```

Figure 7: Setting up memory

With EX6 in memory, RESEARCHER process the same noun phrase, "a motor spindle" as shown in Figure 8. We can see that when it queries memory to try and find known relations between motors and drive shafts, it finds the one from EX6 and assumes it to hold here, as well. Thus, an "on top of" relation is added to the representation, showing the genuinely dynamic nature of RESEARCHER's text processing.

```
Running RESEARCHER at 3:18:11 PM

(A MOTOR SPINDLE)

Processing:

A              : New instance word -- skip
MOTOR          : Memette within NP; save and skip
SPINDLE        : NP word -- memette DRIVE-SHAFT#
  New DRIVE-SHAFT# instance (&MEM3)
  >>> Looking for relation between MOTOR# and &MEM3 (DRIVE-SHAFT#)
  New MOTOR# instance (&MEM4)
  Establishing R-ON-TOP-OF; SUBJECT: &MEM4 (MOTOR#); OBJECT: &MEM3 (DRIVE-SHAFT#) [&REL2]

Text Representation:

----------------------B-3   3 = DRIVE-SHAFT#

----------------------B-4   4 = MOTOR#

A list of relations:

           Subject:              Relation:          Object:

[&REL2/B]  &MEM4 (MOTOR#)        (R-ON-TOP-OF)      &MEM3 (DRIVE-SHAFT#)
```

Figure 8: 'Spindle motor' with EX6 in memory

As a further illustration of RESEARCHER's use of memory in text processing, we will show how it processes part of a real patent abstract, EX7.

EX7 - P58; United States Patent #4287445 (abstract)

An electromagnetic linear actuator for positioning a transducer over locations on a rotating magnetic recording disk comprising an actuator housing used as a stationary base for supporting various parts; a coil and cart assembly including a cart, having a rectangular cross section and tubular in construction, adapted at one end to support the transducer ...

Although it may not be immediately obvious, the beginning of EX7 is extremely ambiguous. (It may not be obvious because people are so good at resolving ambiguity.) The internal structures of the various noun phrases and the determination of what is a part of what could all be resolved in several ways. Without any information in memory, RESEARCHER would have to rely on general heuristics which might or might not work, but would certainly be quite *ad hoc*. Instead, we will provide RESEARCHER with a few (admittedly somewhat artificial) examples that it can use. Specifically, we will give it the following descriptions:

A disc drive comprised of an actuator that has a housing; an assembly with a coil.

A cylindrical cart with one modified end and a rectangular cross section.

Having given RESEARCHER examples of support mechanisms and double sided floppy disc drives, we let it read EX6. The first part of processing is shown in Figure 9.

A number of aspects of RESEARCHER's text processing are shown in Figure 9. We will focus on its use of memory. Each memory access is again indicated by ">>>". The beginning of the processing of P58 is relatively simple. As we can see in Figure 9 how RESEARCHER processes the first noun group with a "skip and save" strategy [Lebowitz 83e]. The words "electromagnetic" and "linear" are saved until the head noun, "actuator" is reached. It then works back through the noun group, applying the modifiers to the actuator. Next to be established are a purposive relation, P-GUIDES, taken from the word "positioning" (after "for", which in this case indicates that a purpose word is to follow) and a physical

```
Running RESEARCHER at 28-Jul-85 19:37:57
Patent: P58

(AN ELECTROMAGNETIC LINEAR ACTUATOR FOR POSITIONING A TRANSDUCER OVER
LOCATIONS ON A ROTATING MAGNETIC RECORDING DISK COMPRISING AN ACTUATOR
HOUSING USED AS A STATIONARY BASE FOR SUPPORTING VARIOUS PARTS *SEMI* A COIL
AND CART ASSEMBLY INCLUDING A CART *COMMA* HAVING A RECTANGULAR CROSS
SECTION AND TUBULAR IN CONSTRUCTION *COMMA* ADAPTED AT ONE END TO SUPPORT
THE TRANSDUCER *STOP*)

Processing:

AN              : New instance word -- skip
ELECTROMAGNETIC : Memette modifier; save and skip
LINEAR          : Memette modifier; save and skip
ACTUATOR        : MP word -- memette ACTUATOR#
  New ACTUATOR# instance (&MEM8)
  Augmenting &MEM8 (ACTUATOR#) with feature: CONFIGURATION = LINEAR
  Augmenting &MEM8 (ACTUATOR#) with feature: TYPE/PURPOSE = ELECTROMAGNETIC
FOR (FOR1)      : Purpose indicator -- skip
POSITIONING     : Purpose word -- save and skip
A               : New instance word -- skip
TRANSDUCER      : MP word -- memette TRANSDUCER#
  New TRANSDUCER# instance (&MEM9)
  Establishing P-GUIDES relation; SUBJECT: &MEM8 (ACTUATOR#);
    OBJECT: &MEM9 (TRANSDUCER#) [&REL1]
OVER            : Relation word -- save and skip
LOCATIONS       : MP word -- memette LOCATION#
  New LOCATION# instance (&MEM10)
  >>> Refining R-ABOVE OBJECT from &MEM9 (TRANSDUCER#) &MEM8 (ACTUATOR#)
  Establishing R-ABOVE relation; OBJECT: &MEM9 (TRANSDUCER#);
    SUBJECT: &MEM10 (LOCATION#) [&REL2]
ON (ON2)        : Part of indicator
  Assuming &MEM10 (LOCATION#) or &MEM9 (TRANSDUCER#) or &MEM8 (ACTUATOR#)
   is part of the following
A               : New instance word -- skip
ROTATING        : Purpose word within MP; save and skip
MAGNETIC        : Memette modifier; save and skip
RECORDING       : Purpose word within MP; save and skip
DISK            : MP word -- memette DISC#
  New DISC# instance (&MEM11)
  Establishing P-WRITES relation; OBJECT: &MEM11 (DISC#) [&REL3]
  Augmenting &MEM11 (DISC#) with feature: TYPE/PURPOSE = MAGNETIC
  Establishing P-ROTATES relation; OBJECT: &MEM11 (DISC#) [&REL4]
  >>> Selecting comp for &MEM11 (DISC#) from among &MEM10 (LOCATION#)
        &MEM9 (TRANSDUCER#) &MEM8 (ACTUATOR#)
  Assuming &MEM10 (LOCATION#) is part of &MEM11 (DISC#)
```

Figure 9: RESEARCHER processing the first part of P58

relation, R-ABOVE from the word "over". In establishing the R-ABOVE relation, RESEARCHER must decide whether the "transducer" or the "actuator" is over the "locations". There being no relevant information in memory, and the objects being similar from the point of view of the system's heuristics, the most recent is picked.

RESEARCHER's processing gets more interesting when the noun group "a rotating magnetic recording disk" is reached. The processing begins similarly to that for the first noun group, saving the modifiers and then applying them to disc#. But here RESEARCHER must decide whether the "part of"

word, "on", indicates that "locations" or the "actuator" is part of the disk. (If the second reading is not obvious, imagine the word "and" before "on". This reading is syntactically possible, with or without the word "and" being present.)

RESEARCHER's first choice of how to resolve this ambiguity is with a memory check. It looks for cases in memory where either the concept location# or actuator# is part of disc#. Finding none, since we have only provided the system with a simple memory for this example, RESEARCHER resorts to its set of heuristics. The relevant rule states that "virtual" objects, such as location#, which refer to implicit parts of objects, are more likely to be parts of solid objects (such as disk#) than are complex objects (such as actuator#).[5] This sort of processing is related to the use of semantic properties of words for disambiguation, and is integrated nicely with memory search. However, we wish to avoid adding too many *ad hoc* rules of this sort -- the system currently has about six such rules which seem to cover most of the cases where memory is unavailable.

More specific use of memory by RESEARCHER occurs in the processing of the next section of P58, shown in Figure 10.

The first noun group processed in Figure 10, "an actuator housing", includes a noun-noun construction requiring memory access. RESEARCHER must determine the relationship between the two objects described, actuator# and enclosure# ("housing"). There are no syntactic clues or semantic clues as to the relation. So, RESEARCHER goes to memory, finds the assembly/part relation that exists in memory for other instance of these concepts (from an example we gave it), and assumes that this relation holds for the new example. Had there been a more complex construction, say noun-noun-noun, RESEARCHER would have used this same information in memory to determine which objects related to each other.

[5]Other such virtual objects are side#, top#, etc.

```
COMPRISING     : Parts of &MEM11 (DISC#) or &MEM10 (LOCATION#) or &MEM9 (TRANSDUCER#)
                   or &MEM8 (ACTUATOR#) to follow
AN             : New instance word -- skip
ACTUATOR       : Memette within NP; save and skip
HOUSING        : NP word -- memette ENCLOSURE#
  New ENCLOSURE# instance (&MEM12)
  >>> Looking for relation between ACTUATOR# and &MEM12 (ENCLOSURE#)
  Assuming &MEM12 (ENCLOSURE#) is part of &MEM8 (ACTUATOR#)
  >>> Selecting assy for &MEM12 (ENCLOSURE#) from among &MEM11 (DISC#)
        &MEM10 (LOCATION#) &MEM9 (TRANSDUCER#) &MEM8 (ACTUATOR#)
  &MEM12 (ENCLOSURE#) is already known to be a part of &MEM8 (ACTUATOR#)
USED AS        : Phrase
-> USED-AS     : Purpose word -- save and skip
A              : New instance word -- skip
STATIONARY     : Memette modifier; save and skip
BASE           : NP word -- memette BASE#
  New BASE# instance (&MEM13)
  Augmenting &MEM13 (BASE#) with feature: MOBILITY = NONE
  Assuming &MEM13 (BASE#) is part of &MEM8 (ACTUATOR#)
  >>> Refining P-ACTS-AS SUBJECT from &MEM12 (ENCLOSURE#) &MEM8 (ACTUATOR#)
  Establishing P-ACTS-AS relation; SUBJECT: &MEM12 (ENCLOSURE#);
    OBJECT: &MEM13 (BASE#) [&REL5]
FOR (FOR1)     : Purpose indicator -- skip
SUPPORTING     : Purpose word -- save and skip
VARIOUS        : Memette modifier; save and skip
PARTS          : NP word -- memette PART#
  New PART# instance (&MEM14)
  Augmenting &MEM14 (PART#) with feature: NUMBER = SOME
  Assuming &MEM14 (PART#) is part of &MEM8 (ACTUATOR#)
  >>> Refining R-CONNECTED-TO SUBJECT from &MEM13 (BASE#) &MEM12 (ENCLOSURE#)
  Unable to select SUBJECT -- using most recent
  Establishing R-CONNECTED-TO relation; SUBJECT: &MEM13 (BASE#);
    OBJECT: &MEM14 (PART#) [&REL6]
  Establishing P-SUPPORTS relation; SUBJECT: &MEM13 (BASE#); OBJECT: &MEM14 (PART#) [&REL7]
*SEMI*         : Skip (SKIP)
```

Figure 10: RESEARCHER processing the second part of P58

The same assembly/part relation in memory is used to resolve another textual ambiguity. RESEARCHER must determine whether the actuator or the disk "comprises" the "actuator housing". Again, the existing relation in memory resolves this ambiguity, and determines that the housing is part of the actuator. Since this relationship has already been established (while analyzing the noun group) processing simply moves on.

The remainder of Figure 10 shows more examples of RESEARCHER identifying object, relating them to each other and accessing memory to resolve ambiguity. Figure 11 shows further examples of all of these sorts of processing as RESEARCHER completes the first fragment of P58.

One interesting aspect of the processing in Figure 11 is the handling of the phrase "coil and cart assembly". As for the earlier noun-noun constructions, RESEARCHER must determine the relations

```
A               : New instance word -- skip
COIL            : Memette within NP; save and skip
AND (AND2)      : Skip (SKIP)
CART            : Memette within NP; save and skip
ASSEMBLY        : MP word -- memette UNKNOWN-ASSEMBLY#
  New UNKNOWN-ASSEMBLY# instance (&MEM15)
  >>> Looking for relation between CARRIAGE# and &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
  New CARRIAGE# instance (&MEM16)
  Assuming &MEM16 (CARRIAGE#) is part of &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
  >>> Looking for relation between COIL# and one of &MEM16 (CARRIAGE#)
        &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
New COIL# instance (&MEM17)
Assuming &MEM17 (COIL#) is part of &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
Assuming &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') is part of &MEM8 (ACTUATOR#)
INCLUDING       : Parts of &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
                    or &MEM8 (ACTUATOR#) to follow
A               : New instance word -- skip
CART            : MP word -- memette CARRIAGE#
  Reference for CARRIAGE#: &MEM16
  >>> Selecting assy for &MEM16 (CARRIAGE#) from among
        &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM8 (ACTUATOR#)
  &MEM16 (CARRIAGE#) is already known to be a part
    of &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
*COMMA*         : Skip (SKIP)
HAVING          : Parts of &MEM16 (CARRIAGE#) or &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY')
                    or &MEM8 (ACTUATOR#) to follow
A               : New instance word -- skip
RECTANGULAR     : Memette modifier; save and skip
CROSS SECTION   : Phrase
-> CROSS-SECTION : MP word -- memette CROSS-SECTION#
  New CROSS-SECTION# instance (&MEM18)
  Augmenting &MEM18 (CROSS-SECTION#) with feature: SHAPE = RECTANGULAR
  >>> Selecting assy for &MEM18 (CROSS-SECTION#) from among &MEM16 (CARRIAGE#)
        &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM8 (ACTUATOR#)
  Assuming &MEM18 (CROSS-SECTION#) is part of &MEM16 (CARRIAGE#)
AND (AND2)      : Skip (SKIP)
TUBULAR         : Memette modifier; save and skip
IN CONSTRUCTION : Phrase
-> IN-CONSTRUCTION : Collecting modifiers
  >>> Looking for memette modified by CONFIGURATION/CYLINDRICAL from
        &MEM18 (CROSS-SECTION#) &MEM16 (CARRIAGE#)
        &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM8 (ACTUATOR#)
  Augmenting &MEM16 (CARRIAGE#) with feature: CONFIGURATION = CYLINDRICAL
*COMMA*         : Skip (SKIP)
ADAPTED         : Purpose word -- save and skip
AT              : Relation word -- save and skip
ONE             : Memette modifier; save and skip
END             : MP word -- memette END#
  New END# instance (&MEM19)
  Augmenting &MEM19 (END#) with feature: NUMBER = 1
  Assuming &MEM19 (END#) is part of &MEM16 (CARRIAGE#)
  Establishing P-MODIFIES relation; SUBJECT: &MEM19 (END#) [&REL8]
  Establishing R-AT relation; SUBJECT: &MEM19 (END#) [&REL9]
TO SUPPORT      : Phrase
-> SUPPORTS     : Purpose word -- save and skip
THE             : Antecedant word -- skip
TRANSDUCER      : MP word -- memette TRANSDUCER#
  Reference for TRANSDUCER#: &MEM9
  >>> Refining P-SUPPORTS SUBJECT from &MEM19 (END#) &MEM16 (CARRIAGE#)
        &MEM15 (UNKNOWN-ASSEMBLY# -- 'ASSEMBLY') &MEM8 (ACTUATOR#)
  Unable to select SUBJECT -- using most recent
  Establishing P-SUPPORTS relation; SUBJECT: &MEM19 (END#);
    OBJECT: &MEM9 (TRANSDUCER#) [&REL10]
*STOP*          : Break word -- skip
```

Figure 11: RESEARCHER processing the third part of P58

between the "assembly", the "coil" and the "cart". Naturally, RESEARCHER accesses memory in each

case. In one instance, for the "coil", it finds the example existing in memory. In the other case, relating "cart" and "assembly", RESEARCHER must rely on its heuristic that vague, complex objects, like "assemblies", usually contain more specific objects as parts. (Actually, to do this example perfectly, we would also have to apply similar techniques to determine the scope of "and" as a connective.)

Figure 12 gives the representation that RESEARCHER has derived by using memory to help understand P58. The key point is that a correct representation could not be found from syntactic or semantic rules alone. Some form of object memory is needed. While a larger memory would force us to consider more deeply problems of multiple relevant examples, we feel the approach is a sound one.

```
Text Representation:

            |--------E-13                              8 = TYPE/PURPOSE/ELECTROMAGNETIC
            |                                            CONFIGURATION/LINEAR ACTUATOR#
            |------EFG-14                              12 = ENCLOSURE#
            |-------FG-15                              13 = MOBILITY/NONE BASE#
            |                          |----------19   14 = NUMBER/>1 PART#
---------A-9|----------16|----------17|------HIJ-20   15 = UNKNOWN-ASSEMBLY#
            |            |                            16 = CONFIGURATION/CYLINDRICAL CART#
            |            |----------18                17 = COIL#
            |                                         18 = SHAPE/RECTANGULAR CROSS-SECTION#
            |                                         19 = NUMBER/1 END#

------------------ABJ-10   9 = TRANSDUCER#

-------------CD-12|--------------B-11  10 = NUMBER/>1 LOCATION#
                                       11 = TYPE/PURPOSE/MAGNETIC DISC#

A list of relations:

           Subject:              Relation:          Object:

[&REL1/A]  &MEM8  (ACTUATOR#)    (P-GUIDES)         &MEM9  (TRANSDUCER#)
[&REL2/B]  &MEM10 (LOCATION#)    (R-ABOVE)          &MEM9  (TRANSDUCER#)
[&REL3/C]                        (P-WRITES)         &MEM11 (DISC#)
[&REL4/D]                        (P-ROTATES)        &MEM11 (DISC#)
[&REL5/E]  &MEM12 (ENCLOSURE#)   (P-ACTS-AS)        &MEM13 (BASE#)
[&REL6/F]  &MEM13 (BASE#)        (R-CONNECTED-TO)   &MEM14 (PART#)
[&REL7/G]  &MEM13 (BASE#)        (P-SUPPORTS)       &MEM14 (PART#)
[&REL8/H]  &MEM19 (END#)         (P-MODIFIES)
[&REL9/I]  &MEM19 (END#)         (R-AT)
[&REL10/J] &MEM19 (END#)         (P-SUPPORTS)       &MEM9  (TRANSDUCER#)
```

Figure 12: The representation of EX7

We have much left to do in our integration of text processing and memory. However, we feel our general approach is quite promising, as our work in building up memory has a positive synergistic effect on text processing robustness. The identification of specific questions to ask memory seems to be much

more effective than looking for more general applications of memory to understanding or developing huge numbers of *ad hoc* disambiguation rules.

4 User Expertise and Question-Answering

Once a substantial knowledge base has been built up by RESEARCHER, it is important that it can be queried intelligently. In [Lebowitz 83b; Paris 84] we described an early question answering module. Recently, our work has concentrated on how RESEARCHER might tailor its answers to the needs of individual users. There are many elements to such tailoring, the goal of the user, for example, but here we will concentrate on just one factor -- the user's level of expertise. We have tried to determine the sorts of basic answering strategies that would be appropriate for expert and naive users of the system.[6] Eventually, we will also look at how expertise affects other levels of processing (such as lexical choice) as well as other factors that require the tailoring of answers.

In order to get an idea about the kinds of strategies that might be appropriate for various users, we have looked at texts that describe objects and that are aimed at readers with different levels of expertise -- several adult and junior encyclopedias. As described more fully in [Paris 85], the strategies used in adult and junior encyclopedias are quite different -- the adult encyclopedias, presumably aimed at relative experts, tend to describe the part structure of objects, while the junior encyclopedias describe the processes that take place in the device. EX8 and EX9 show this distinction for descriptions of telephones.

> EX8 - The hand-sets introduced in 1947 consist of a receiver and a transmitter in a single housing available in black or colored plastic. The transmitter diaphragm is clamped rigidly at its edges to improve the high frequency response. The diaphragm is coupled to a doubly resonant system -- a cavity and an air chamber -- which broadens the response... (Collier's Encyclopedia, 1962)

As we can see, EX8, taken from an adult encyclopedia, describes a telephone by presenting its

[6] Actually, user expertise falls into two areas – familiarity with the system and familiarity with the domain. We are concerned here with the latter.

parts. The description continues in this vein. It is using a construction quite similar to the constituency schema that McKeown used in her question answering work [McKeown 82], providing an almost tree-like description of the parts of the object. This is in contrast with a description aimed at younger readers, EX9.

> **EX9** - When one speaks into the transmitter of a modern telephone, these sound waves strike against an *aluminum disk or diaphragm* and cause it to vibrate back and forth in just the same way the molecules of air are vibrating... (Britannica Junior, 1963)

Here the description is process-oriented. It traces the process of transmitting sound, introducing part descriptions only when necessary. This is clearly a different presentation strategy, and our study of texts indicates that it is much more widely used in texts aimed at less experienced readers. We feel that a process-oriented answer would be appropriate for RESEARCHER to use when dealing with a novice user not likely to know what various parts are used for.

We are currently implementing these two different strategies for describing the same object. Figure 13 shows how RESEARCHER generates an explanation of a telephone from a hand-coded knowledge base making use of McKeown's constituency schema. This is presumably the kind of explanation we would have RESEARCHER generate for an expert user. In Figure 13 we can see the various steps of the explanation that RESEARCHER produces, annotated by hand on the right. We are currently interfacing RESEARCHER with a surface generator that uses a functional grammar [Kay 79] to automatically produce English output.

Figure 14 shows the first part of another description of a telephone generated by RESEARCHER from the same knowledge base, this time in a process-based manner appropriate for a novice. Instead of just giving the parts of the telephone, it now describes the operation of the device, primarily by presenting a series of causal links. Our surface generator under development will also produce English text for this example.

The manner in which RESEARCHER generates the descriptions in Figure 13 and Figure 14 is

```
; Description of the TELEPHONE based on the Constituency schema.

; &MEMX are the unique identifiers for the object frames.
; The Constituency Schema was filled by stepping through an ATN.

&MEM1  (TELEPHONE)                                          ; The telephone is
       (*IDENTIFICATION* (VARIANT-OF: DEVICE#))             ; a device.  It consists
       (*CONSTITUENCY* (&MEM2 (TRANSMITTER))                ; of a transmitter,
                       (&MEM16 (HOUSING))                   ; a housing, a line and
                       (&MEM15 (LINE))                      ; a receiver.
                       (&MEM17 (RECEIVER)))

&MEM2  (TRANSMITTER)                                        ; The transmitter is
       (*IDENTIFICATION* (VARIANT-OF: TRANSMITTER#))        ; a kind of transmitter.
       (*CONSTITUENCY* (&MEM6 (DOUBLY-RESONANT-SYSTEM));    It has a doubly
                       (&MEM3 (DIAPHRAGM-T)))               ; resonant system and
                                                            ; a diaphragm;
&MEM16 (HOUSING)                                            ; the housing is
       (*IDENTIFICATION* (VARIANT-OF: COVER#))              ; a type of cover;
       (*CONSTITUENCY*)

&MEM5  (LINE)                                               ; the line is a wire;
       (*IDENTIFICATION* (VARIANT-OF: WIRE#))
       (*CONSTITUENCY*)

&MEM17 (RECEIVER)                                           ; The receiver is a
       (*IDENTIFICATION* (VARIANT-OF: RECEIVER#))           ; kind of receiver.
       (*CONSTITUENCY* (&MEM22 (DIAPHRAGM-T))               ; It consists of a
                       (&MEM21 (AIR-GAP))                   ; diaphragm, an air gap
                       (&MEM18 (ELECTROMAGNET)))            ; and an electromagnet.
```

Figure 13: A constituency explanation of 'telephone'

described in more detail in [Paris 85]. In addition to looking at the different generation strategies, we are also studying ways to determine the expertise of a user, since simply asking is not always the appropriate approach, particularly as a user's expertise may vary over different topics of discussion. We are also considering mixed strategies that make use of elements of each of the generation techniques illustrated here.

5 Conclusion

We have described here three areas of investigation in the study of intelligent information systems, involving the prototype system RESEARCHER. Our work involves the basic idea of *understanding* the text to be stored in the system's memory. Remarkably little work in Artificial Intelligence has taken this approach. The system CyFr [Schank et al. 80], which combined FRUMP [DeJong 79] and CYRUS [Kolodner 84], worked from a similar perspective as did our earlier work with IPP [Lebowitz 83c], a system that read news stories about terrorism. Other work applying Artificial Intelligence to Information Retrieval, such as [Tou et al. 82; Tong, et al. 83], has either applied heuristic approaches to the search of

```
; The process information gets picked up and printed out for a naive  user.
; &MRX are the unique identifiers to the frames corresponding to the
; meta-relations the program is tracing.

&REL3 (P-SPEAKS-INTO):                      ; When one speaks into the
        subject : (&MEM27) [ONE]            ; transmitter of a telephone,
        object  : (&MEM2) [TRANSMITTER]

    ===>   &MR0 (M-CAUSES)

&REL4 (P-HITS):                             ; the sound waves  hit
        subject : (&MEM28) [SOUNDWAVES]     ; the diaphragm of the transmitter.
        object  : (&MEM3) [DIAPHRAGM-T]

 --------

&REL4 (P-HITS):                             ; This causes
        subject : (&MEM28) [SOUNDWAVES]     ;
        object  : (&MEM3) [DIAPHRAGM-T]

    ===>   &MR1 (M-CAUSES)

&REL5 (P-VIBRATES):                         ; the diaphragm to vibrate
         subject :
         object  : (&MEM3) [DIAPHRAGM-T]

 --------

&REL5 (P-VIBRATES):
         subject :
         object  : (&MEM3) [DIAPHRAGM-T]

    ===>   &MR2 (M-EQUIVALENT-TO)           ; in the same manner as

&REL8 (P-VIBRATES)                          ; the molecules of air
         subject :                          ; are vibrating.
         object  : (&MEM26) [AIR-MOLECULES]
```

Figure 14: Process-oreiented description of 'telephone'

raw text or involved search through carefully prepared knowledge bases. Understanding the text greatly widens the possibilities. In our own research, the generalization of hierarchical representations allows the system to learn about a wide range of complex objects and build up a rich memory. This memory is used extensively in text processing, primarily for disambiguation, to achieve robust performance. Finally, awareness of the expertise level of a user will allow RESEARCHER to tailor it answers to each user. The sum of these three related areas of investigation should lead towards the development of powerful intelligent information systems that can make better use of huge amounts of varying sorts of information than can purely text-based systems.

References

[Abelson 73] Abelson, R. P. The structure of belief systems. In R. C. Schank and K. Colby, Ed., *Computer Models of Thought and Language*, W. H. Freeman Co., San Francisco, 1973.

[Barr et al. 82] Barr, A., Cohen, P. R. and Feigenbaum, E. A., eds. *The Handbook of Artificial Intelligence, Volumes 1 - 3.* William Kaufmann, Inc., Los Altos, California, 1982.

[Birnbaum and Selfridge 81] Birnbaum, L. and Selfridge, M. Conceptual analysis of natural language. In R. C. Schank and C. K. Riesbeck, Ed., *Inside Computer Understanding*, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1981, pp. 318 - 353.

[Carbonell 81] Carbonell, J. G. *Subjective Understanding: Computer Models of Belief Systems.* UMI Research Press, Ann Arbor, Michigan, 1981.

[DeJong 79] DeJong, G. F. "Prediction and substantiation: A new approach to natural language processing." *Cognitive Science 3*, 1979, pp. 251 - 273.

[DeJong 83] DeJong, G. F. Artificial intelligence implications for information retrieval. Proceedings of the Sixth International ACM SIGIR Conference, ACM SIGIR, Washington, DC, 1983.

[Evans 68] Evans, T. G. A heuristic program to solve geometric analogy problems. In M. Minsky, Ed., *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968.

[Finin 82] Finin, T. W. The interpretation of nominal compounds in discourse. Technical Report MS-CIS-82-3, Moore School of Engineering, University of Pennsylvania, 1982.

[Harris 78] Harris, L. R. Natural language processing applied to data base query. Proceedings of the 1978 ACM Annual Conference, Association for Computer Machinery, Washington, D. C., 1978.

[Heaps 78] Heaps, H. S. *Information Retrieval: Computational and Theoretical Aspects.* Academic Press, New York, 1978.

[Hirst 83] Hirst, G. *Semantic interpretation against ambiguity.* Ph.D. Thesis, Department of Computer Science, Brown University, 1983.

[Kay 79] Kay, M. Functional grammar. Proceedings of the Fifth Meeting of the Berkeley Linguistics Society, Berkeley, CA, 1979.

[Kolodner 84] Kolodner, J. L. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[Lebowitz 81] Lebowitz, M. Cancelled due to lack of interest. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981.

[Lebowitz 82] Lebowitz, M. "Correcting erroneous generalizations." *Cognition and Brain Theory 5*, 4, 1982, pp. 367 - 381.

[Lebowitz 83a] Lebowitz, M. Intelligent information systems. Proceedings of the Sixth International ACM SIGIR Conference, ACM SIGIR, Washington, DC, 1983, pp. 25 - 30.

[Lebowitz 83b] Lebowitz, M. RESEARCHER: An overview. Proceedings of the Third National Conference on Artificial Intelligence, Washington, DC, 1983, pp. 232 - 235.

[Lebowitz 83c] Lebowitz, M. "Generalization from natural language text." *Cognitive Science 7*, 1, 1983, pp. 1 - 40.

[Lebowitz 83d] Lebowitz, M. Concept learning in a rich input domain. Proceedings of the 1983 International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 177 - 182. To appear in

Machine Learning: An Artificial Intelligence Approach, Volume II.

[Lebowitz 83e] Lebowitz, M. "Memory-based parsing." *Artificial Intelligence 21*, 4, 1983, pp. 363 - 404.

[Lebowitz 84] Lebowitz, M. Using memory in text understanding. Proceedings of ECAI-84, Pisa, Italy, 1984.

[Levi 78] Levi, J. N. *The Syntax and Semantics of Complex Nominals.* McGraw Hill, New York, 1978.

[McKeown 82] McKeown, K. R. *Generating natural language text in response to questions about database structure.* Ph.D. Thesis, University of Pennsylvania, 1982.

[Michalski 80] Michalski, R. S. "Pattern recognition as rule-guided inductive inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence 2*, 4, 1980, pp. 349 - 361.

[Paris 84] Paris, C. L. Determining the level of expertise. Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing, Atlanta, Georgia, 1984.

[Paris 85] Paris, C. L. Description strategies for naive and expert users. Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, 1985.

[Salton and McGill 83] Salton, G. and McGill, M. J. *Introduction to Modern Information Retrieval.* McGraw Hill, New York, 1983.

[Schank 79] Schank, R. C. "Interestingness: Controlling inference." *Artificial Intelligence 12*, 3, 1979, pp. 273 - 297.

[Schank et al. 80] Schank, R. C., Kolodner, J. L. and DeJong, G. F. Conceptual information retrieval. Technical Report 190, Yale University Department of Computer Science, 1980.

[Small 80] Small, S. Word expert parsing: A theory of distributed word-based natural language understanding. Technical Report TR-954, University of Maryland, Department of Computer Science, 1980.

[Tong, et al. 83] Tong, R. M., Shapiro, D. G., McCune, B. P. and Dean, J. S. A rule-based approach to information retrieval: Some results and comments. Proceedings of the Third National Conference on Artificial Intelligence, Washington, DC, 1983.

[Tou et al. 82] Tou, F. N., Williams, M. D., Fikes, R., Henderson, A., and Malone, T. RABBIT: An intelligent database assistant. Proceedings of the Second National Conference on Artificial Intelligence, Pittsburgh, PA, 1982, pp. 314 - 318.

[Wasserman 84] Wasserman, K. Understanding hierarchically structured objects. Columbia University Department of Computer Science, 1984.

[Wasserman 85] Wasserman, K. *Unifying representation and generalization: Understanding hierarchically structured objects.* Ph.D. Thesis, Columbia University Department of Computer Science, 1985.

[Wasserman and Lebowitz 83] Wasserman, K. and Lebowitz, M. "Representing complex physical objects." *Cognition and Brain Theory 6*, 3, 1983, pp. 333 - 352.

[Winston 72] Winston, P. H. Learning structural descriptions from examples. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1972.

[Winston 80] Winston, P. H. "Learning and reasoning by analogy." *Communications of the ACM 23*, 1980, pp. 689 - 702.