Disambiguating Prepositional Phrase Attachments by Using On-Line Dictionary Definitions

Karen Jensen and Jean-Louis Binot*

Computer Sciences Department
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

Standard on-line dictionaries offer a wealth of knowledge expressed in natural language form. We claim that such knowledge can and should be accessed by natural language processing systems to solve difficult ambiguity problems. This paper sustains that claim by describing a set of computational tools and techniques used to disambiguate prepositional phrase attachments in English sentences, by accessing on-line dictionary definitions. Such techniques offer hope for eliminating the time-consuming, and often incomplete, hand coding of semantic information that has been conventional in natural language understanding systems.

1. Introduction

A customary pattern for papers on natural language processing runs roughly as follows:

1. Here's a difficult language situation.
2. Here's the semantic (pragmatic / discourse / whatever) information necessary to interpret the situation as we humans interpret it.
3. Here's how I put this information into my system.
4. Therefore, my system can handle the situation.

Although the amount of effort which has been expended in doing this kind of thing is admirable, there is still something less than totally satisfactory about the approach. First, it tends to trivialize the notion of "solution to a problem." Second, and more important, the approach relies on hand-coding pieces of information (commonly called "knowledge"). The more we want our programs to know, the more we have to hand-feed them. We can keep on doing this, and maybe, after several generations, someone will put all the pieces together and discover that AI researchers have succeeded in rewriting the history of the universe as we understand it.

Without question, this will be a form of success. But there might be a faster way to the goal —one that doesn't involve reinventing the wheel of knowledge. We already have volumes of hand-coded natural language information in our reference books —dictionaries and encyclopedias, for instance. If computers could access that information, and use it to help get out of difficult situations, they (and we) would be much further ahead. Problems should be expected, of course: reference works are arbitrary and inconsistent; just having the information does not guarantee being able to use it.

It is nevertheless possible to think of on-line reference books as knowledge bases. In this paper we propose techniques for processing the definitions of an on-line standard dictionary, and for extracting from them the semantic information necessary to resolve the ambiguities inherent in the attachment of English prepositional phrases. We consult the on-line dictionary as if it were a semantic expert, and we find in it the kind of information that has previously been supplied by means of scripts, frames, templates, and similar hand-crafted devices.

We start by presenting PEG, a broad-coverage computational grammar of English which is our tool for analyzing on-line definitions, and by discussing the tools necessary for extracting from these definitions the knowledge relevant for disambiguation. Subsequent sections describe these tools in greater detail, focussing on specific sentences and on the heuristic machinery used to discover their most likely interpretations. A final section sums up the work and suggests some

* The second author currently works for B.I.M., Belgium.

0362-613X/87/030251-260$03.00

important areas for future research. Two appendices provide traces of the processing of some examples, and technical information about the "approximate reasoning" techniques used in our system.

2. PEG and Syntactic Parsing

The computational grammar called PEG (PLNLP English Grammar) analyzes English sentences by using syntactic information in augmented phrase structure rules with a bottom-up, parallel processor. The system that provides these facilities is PLNLP, or the Programming Language for Natural Language Processing (Heidorn 1975). PEG produces useful sentence parses for a large amount, and a wide variety, of English text. These parses contain syntactic and functional (subject, object, etc.) information, but, as yet, no semantic information, or other information beyond the functional level.

We do not argue against the general notion of integrating syntax and semantics. When natural language processing techniques are further developed, we will probably be able to see more clearly how this integration should best be accomplished. But there are also independent reasons for postponing the use of semantics as long as possible. And the development of PEG has shown that a syntax-only stage of parsing is not computationally expensive, and is both efficient and useful (Jensen 1986).

There is an enduring notion that allowing ambiguities to persist in a syntactic parser must cause a disastrous combinatorial explosion in the number of parses produced. This is not true. In the first place, syntax is more helpful than has generally been supposed, and can be used to filter out the grossly unlikely parses. In the second place, there are economical ways of dealing with ambiguities. PEG's method, during the syntactic phase of parsing, is to attach pre- and post-modifiers in a single arbitrary pattern (usually to the closest possible head, which is always labeled with "*"), but to mark other possible attachment sites so that they can be referred to for later semantic processing. A question mark indicates these possible attachment sites:

```
DECL   NP      PRON*   "I"
       VERB*   "ate"
       NP      DET     ADJ*    "a"
               NOUN*   "fish"
        ?      PP      PREP    "with"
                       DET     ADJ*    "a"
                       NOUN*   "fork"
       PUNC    "."
```

Figure 1. PEG parse tree for a syntactically ambiguous sentence, "I ate a fish with a fork."

This is a compact shorthand for expressing attachment ambiguities. The attachment strategy eliminates any combinatorial explosion; it does not significantly affect parsing time; and it allows for the full range of later semantic processing.

One of the jobs that needs doing in order to move beyond syntax is the elimination of syntactically legal, but semantically anomalous, ambiguities like the ambiguity in Figure 1. All sorts of modifiers can potentially be multiply attached: nouns, adjectives, adverbs, phrases, and clauses. In this paper we will concentrate on the disambiguation of prepositional phrase attachments, which certainly qualifies as a difficult problem in natural language processing, and also one which is of considerable current interest (Lytinen 1986, Dahlgren and McDowell 1986, Schubert 1986).

3. Dictionary Definitions

One of the strengths of PEG is that it is a genuinely broad-coverage grammar; it can attempt a syntactic parse of any sentence in ordinary English. This is a gain, and should not be lost as we move into semantic processing. Having a largely domain-independent syntax, we do not want to create a domain-dependent semantics.

There is, of course, one repository of semantic information which already exists, which has been developed over many years, and which applies in a broad fashion to almost all English words, and to large subsets of the vocabulary of all task domains. This is the defining of words and word senses in a good standard dictionary. We have been using syntactic information from two on-line dictionaries: Webster's Seventh New Collegiate (W7) and the Longman Dictionary of Contemporary English (LDOCE). The next step is to explore on-line definitions with the goal of extracting semantic information. For this initial stage of research, we have used only W7.

Dictionary definitions contain the following kinds of information that are of interest to the present task:

- the definition(s);
- example sentences and phrases;
- synonyms;
- usage notes and other comments.

In order to make use of this information for natural language processing, we need at least:

- a tool to parse dictionary definitions and example sentences;
- a tool to extract the semantically relevant information from the definitions and examples;
- a tool for looking up the definitions of related words: superordinate words or ancestors (hypernyms), subordinate words or descendents (hyponyms), and same-level words (synonyms and antonyms);
- the robustness necessary to cope with the many potential errors or format variations which can appear in a man-made dictionary.

PEG provides the necessary parsing tool. Robust tools for extracting information from dictionary entries, and

for making use of that information to disambiguate attachment problems, are described in the following sections.

4. Heuristics

One of the key factors for achieving robustness is the recognition that every single rule that we might want to write for processing dictionary entries is bound to fail at least once in a while. Therefore rules should be designed as **heuristics,** which can only provide approximate results. The cumulative effect of many heuristics, and not the perfection of each one taken separately, has to do the job.

As in a typical "approximate reasoning" paradigm, all heuristics should only yield weighted preferences, or **certainty factors,** and no categorical acceptance or rejection of an interpretation. Thus different heuristics can interact in an additive way, and unforeseen examples will be more easily processed. In that way also, the whole system will be incremental: we can start modestly and progressively augment its capabilities. The process should be fail-proof in the sense that it should always give an answer: at one extreme, this answer should be to state that no better ordering can be made, and that things should be left as they are.

As a first step, we have started to study the heuristics that will be necessary to disambiguate the attachments in sentences using the preposition *with*, and the relationships signaled by this preposition. According to a standard handbook on prepositions, *with* can signal: place, time, agency or instrument, cooperation, opposition, possession or characteristic, separation or alienation, association, and more (Funk and Wagnalls 1953, pp. 34-35). Heuristics will eventually be necessary for all of these relationships. For present purposes however, we are particularly interested in the relationships of INSTRUMENT and PARTOF. PARTOF involves what has been called "inalienable possession" — possession by nature, not by accident. In these terms, a hand would be PARTOF a body, but a label would not be PARTOF a dress. (Other heuristics will be developed to handle other types of possession.)

The relationships in which we are interested can be illustrated by the following sentences (Binot 1985):

(1) I ate a fish with a fork.
(2) I ate a fish with bones.
(3) I ate a fish with my fingers.

In all cases, the ambiguity resides in the placement of the *with* prepositional phrase, which can modify *fish* or *ate*. And in all cases, PEG has the PP attached to the closest possible head, *fish*, with a question mark showing where the PP would be placed if it modified the main verb *ate* (see Figure 1).

The rest of this section presents, in an informal way, our strategy for solving such ambiguities, and the motivations behind some particular heuristics that we are using. The remaining sections provide more details about the implementation of the system, and about difficulties that still have to be resolved.

Focussing on (1) and following the "approximate" approach stated above, another way to phrase the key question is "Is it *more likely* that a fork is associated with a fish or with an act of eating?" To answer that question, the system evaluates separately the plausibility of the two proposed constructs:

(4a) eat with fork
(4b) fish with fork

then orders the solutions, and picks the one with the highest rating.

In the heuristics we are currently using, one construct is rated higher than another if the words that it contains, and their relationship to each other, can be more easily validated in dictionary definitions. For example, we look up the noun *fork* in W7. If, by chance, its definition were to include the phrase *eat with a fork*, then (4a) would be immediately validated. Most likely, however, these exact same words will not occur. But, as we will see, the definition of *fork* contains the phrase *used for taking up*, and *eating* is defined as a kind of *taking* in the dictionary. If we can provide a way to establish these relationships, then (4a) is validated, although somewhat less easily.

Ease (or certainty) of validation becomes progressively less, the harder it is to establish these inter-word relationships. And the relationships are established:

(a) by identifying equivalent function-word patterns in the definitions, such as the equivalence of *used for* and the instrumental *with*;
(b) by linking important definition words (i.e., central terms in definitional phrases, such as heads of phrases, or else synonyms). This can be done by parsing the definitions, identifying the central word(s), and then following hierarchical chains of definitions through the dictionary.

The two main heuristics that will be used to evaluate the plausability of (4a) against (4b) can be described in natural language as follows (for more details on the formalization and implementation of the heuristics, see section 6):

H1- for checking for an INSTRUMENT relation between a head and a *with* complement:
1. if the head is not a verb, the relation doesn't hold (certainty factor = -1);
2. if some "instrument pattern" (see below) exists in the dictionary definition of the complement, and if this pattern points to a defining term that can be linked with the head, then the relation probably holds (certainty factor = 0.7);
3. else assume that there is more chance that the relation doesn't hold (certainty factor = -0.3).

H2- for checking for a PARTOF relation between a head and a *with* complement:
1. if the head is not a noun, the relation doesn't hold (certainty factor = -1);
2. if some "part-of pattern" (see below) exists in the

dictionary definition of the complement, and if this pattern points to a defining term that can be linked with the head, then the PARTOF relation probably holds (certainty factor = 0.7);

3. else assume that there is more chance that the relation doesn't hold (certainty factor = -0.3).

Heuristic answers are expressed in terms of certainty factors which, as in the MYCIN system (Shortliffe 1976), take their values in the range (-1,+1):

-1 expresses absolute disbelief;

0 expresses complete uncertainty;

1 expresses absolute belief.

Intermediate values express varying degrees of belief or disbelief. It must be understood that each certainty factor refers to the specific proposition (or goal) to which the heuristic is applied. Thus, if clause 3 of heuristic H2 is used when applied to the proposition (4b), the resulting certainty factor -0.3 will indicate a relatively moderate disbelief in this proposition, stemming from the fact that the system has not been able to find any positive evidence in the dictionary to sustain it.

The above heuristics are possible because there are specific words and/or phrases in dictionary definitions, forming what we shall call here **patterns**, which are almost systematically used to express specific semantic relations (Markowitz et al. 1986). For the two relations considered here, some of these patterns are:

INSTRUMENT: for, used for, used to, a means for, etc.

PARTOF: part of, arises from, end of, member of, etc.

These prepositional patterns generally take, as their objects, some central term (or terms) in the definition of the complement word. An attempt can then be made to link that term with the head of the construct that is being studied.

Focussing again on example sentence (1), the system starts by examining the first construct, (4a). It parses the definition of the complement *fork*, and discovers at least one INSTRUMENT pattern, *used for*:

fork: 1. An implement with two or more prongs *used esp for taking up* (as in eating), *pitching* or *digging*.

Taking the conjunction into account, the system finds three possible terms: *taking up*, *pitching*, and *digging*, which it tries to link with *eat*. (For the present, we deliberately avoid the phrase *as in eating* which offers a direct match — in order to show that our approach does not rely on such lucky coincidences.) Following hierarchical chains of definitions by parsing one definition and then extracting, for example, its head word, the system is able to establish that *eat* is a direct hyponym of *take* according to W7; the link is thus probably established.

The system then moves on to consider (4b). Since no PARTOF pattern can be found in the definitions of *fork*, this second possible construct will be ranked as much less likely—(4a) receives a certainty factor of +0.7, but (4b) gets a certainty factor of only -0.3. Therefore the system recommends attaching the prepositional phrase to the main verb in sentence (1).

For sentence (2), the constructs to be compared are *eat with bones* and *fish with bones*. In the definition of *bone*, no useful INSTRUMENT pattern is found; so *eat with bones* cannot be easily validated. On the other hand, the first definition of *bone* gives the following PARTOF pattern:

bone: 1. One of the hard *parts of the skeleton of a vertebrate*.

This yields two possible links for *fish*: *skeleton* and *vertebrate*. *Fish* can be identified as a direct hyponym of *vertebrate* according to W7. Therefore, *fish with bones* receives a higher certainty factor than *eat with bones*, and the system recommends attaching the prepositional phrase to the direct object NP in sentence (2).

The word-linking operation, which is used in heuristics H1 and H2, will presumably be used in many more heuristics as we extend the system. Such often-used operations can be made into **subgoals,** for which we can design specific heuristics. The heuristic currently used for establishing links between words is:

H3- for checking if one word (typically the head of a construct to be tested) is linked to some other word (typically a term pointed to by a pattern in a dictionary definition):

1. if head and term have the same base, the link is certainly established (certainty factor = 1);
2. if the term appears as an hyponym or hypernym of the head by following hierarchical patterns in the dictionary, the link is probably established (certainty factor = 0.7);
3. if term and head appear as common hyponyms of the same genus word by following hierarchical patterns in the dictionary, the link is (less) probably established (certainty factor = 0.5);
4. else, the link is probably not established (certainty factor = -0.7).

The link-checking operation embodied in H3 can be likened to the preference-checking operation proposed by Wilks (1975). An established link will validate a kind of semantic restriction which can fail, but which, if satisfied, will increase our confidence in the likelihood of the proposed relation. However, in our case, the information used to check the satisfaction of the preference is not obtained from any hand-made semantic classification, frame, script, or other such structure, but from the W7 dictionary itself. Moreover, it should be pointed out here that we want to take the notion of "preference satisfaction" in a very loose sense. Although the sentences discussed above exemplified only cases of simple inclusion, what we propose to do in the long term is to check if *some connection* can be established between the two terms — the word being defined and the preference indication — by using hyponym, hypernym and synonym information extracted from the dictionary.

Many useful disambiguations can be performed by

using the kind of link checking outlined above. The heuristic approach, however, does not limit us to this mechanism; and it would indeed be a mistake to accept such a limit. Sentence (3) is a case in point. Here, no useful INSTRUMENT or PARTOF pattern can be found in the definitions of *finger*. However our strong intuition is that the presence of a possessive my, which disagrees in person with the possible head *fish*, precludes any PARTOF relation. This intuition can be formulated in the following heuristic:

H4- for checking for a PARTOF relationship between a head and a with complement:

1. if
   a. the head is a noun, and
   b. the complement is qualified by a possessive, and
   c. this possessive doesn't agree in person with the head, then the PARTOF relation certainly doesn't hold (certainty factor = -1),
2. else this heuristic doesn't give any indication (certainty factor = 0).

The effect of H4 on sentence (3) will be to give to PARTOF such a bad score that the interpretation *eat with my fingers* will be preferred despite the absence of any positive evidence in its favor. When a possessive does not appear, H4 does not alter, in any way, the results provided by previous heuristics. This is achieved, in the second clause of H4, through the use of the zero certainty factor, which expresses total uncertainty and will not modify any other certainty factor proposed by another heuristic for the same problem. (See Appendix 2 for details on the combining of certainty factors.) Another interesting point illustrated by H4 is the ease with which syntactic and semantic clues can be integrated in this kind of approach. Different heuristics can cover different types of clues, and the resulting certainty factors will be combined to yield a unique, weighted answer to the stated problem.

5. Patterns

One of the most basic capabilities needed to implement the strategy outlined above is the capability to match specific patterns against dictionary definitions. Two problems must be mentioned here:

1. A string level pattern matching mechanism will not do. Consider for example the PARTOF pattern *part of*. What we really need to find is the head of the prepositional phrase introduced by *of* and attached to the noun *part*. This clearly requires us to be able to handle the syntactic structure of the definition.
2. We are — and must be — unsure of the patterns we use. On the one hand, further experience with our system, or other parallel studies, might reveal new useful patterns; on the other hand,it might prove necessary to add new constraints to existing patterns in order to improve their selectivity. We obviously need a *flexible* and *declarative way* to specify patterns, so that changes can always be easily incorporated into the system.

The PEG grammar itself provides an adequate solution to the first problem. Each definition is parsed by PEG, and the pattern-matching is performed on the resulting parse trees. An additional difficulty occurs here. Dictionary definitions rarely form complete sentences. In W7, for example, a definition for a noun is typically formulated as a noun phrase, and a definition for a verb as a verb phrase. PEG has the capability to handle this by setting a switch indicating that the input should parsed as an NP, or as a VP, rather than as a complete sentence. Figure 2 below shows the parse for the first definition of the noun *bone*:

```
NP*     NOUN*   "one"
        PP      PREP    "of"
                DET     ADJ*    "the"
                AJP     ADJ*    "hard"
                NOUN*   "parts"
        ?       PP      PREP    "of"
                        DET     ADJ*    "the"
                        NOUN*   "skeleton"
        ?       ?       PP      PREP    "of"
                                DET     ADJ*    "a"
                                NOUN*   "vertebrate"
```

Figure 2. Parse tree for the first definition of the noun *bone*: *one of the hard parts of the skeleton of a vertebrate.*

To solve the second problem, we have developed a declarative language for specifying patterns to be applied to PEG parse trees. Each node in a parse tree is in fact a PLNLP record with a set of attribute-value pairs. An expression in the pattern language will describe a **target node** by enumerating constraints that this node must satisfy. For example, the pattern:

```
(take head ((segtype pp)
            (has prp ((base of)))
            (in ((has head ((base part)))))))
```

directs the pattern-matcher to find in the parse tree the head of a node of type PP having as value for the attribute PRP (PRePosition) a record representing the preposition *of*, and being a successor in the tree of a node having as head the word *part*. Such patterns are applied by a special function SEARCHPARSE, which takes as arguments a pattern and a parse tree. If called with the pattern shown above and the parse tree of Figure 2, SEARCHPARSE will return as result the following PLNLP record:

```
SEGTYPE   'NOUN'
STR       " skeleton"
SUP       'UNREC'
BASE      'SKELETON'
POS       'ADJ'
POS       'NOUN'
INDIC     SING PERS3
```

Figure 3. PLNLP record for the head of the PP node in the parse tree of Figure 2.

The value of the BASE attribute of that record can then be taken as argument for the link checking mechanism outlined in the last section. The pattern language has a simple, highly recursive structure. It currently offers operations to test the value of an attribute field, of an indicator, of the predecessors and successors of a target node in a tree, and of the position of the string covered by the target node in the input text.

6. Implementation of heuristics.

Heuristics are best stated in some formal declarative language that allows us easily to specify new heuristics. An option would have been to use a classical expert system shell with a built-in inference engine. But it seems unlikely, at least at this stage of the research, that we will need the full power of such an engine, because we will probably not use long reasoning chains; when we need to solve a subgoal, heuristics for that subgoal can be invoked directly by an explicit call in the main heuristic. Thus we decided to dispense with backtracking and with logical unification, and to design our own, simpler but faster, control structures.

The disambiguation system is called via the top-level function CHOOSE, which receives as unique argument a list of **goals**; each goal states one possible interpretation whose plausibility is to be evaluated. As an example, the call corresponding to sentence (1) of section 4 is:

```
(CHOOSE ( (WITH <FISH> <FORK>)
          (WITH <EAT> <FORK>) ))
```

Figure 4. A call to the disambiguation system.

where a name between angled brackets denotes the PLNLP record representing the node having that name as base in the parse tree of the sentence. Thus ⟨EAT⟩ denotes the node labeled "ate" in the parse tree shown in Figure 1.

CHOOSE will return its argument goal list, but sorted in decreasing order of plausibility of each goal. When two or more goals have the same plausibility, the original order will be kept, thereby instructing the system to accept the parse tree created by PEG.

CHOOSE works by applying to each member of its goal list the function SOLVE, which finds all heuristics applicable to a given goal and uses them to determine the plausibility of that goal. As can be seen from Figure 4, the general structure of a goal is:

::= (⟨goal type⟩ ⟨argument⟩*)

— a list starting with a keyword denoting the type of the goal, and then enumerating the specific arguments of the goal. The number and kind of the arguments depend upon the type of goal being considered.

Each heuristic is coded separately as a PLNLP procedure. The following figure gives the formal specification of the heuristic H2, which was informally described in section 4:

```
H2(CONTROLER*PTR,COMPLEMENT*PTR,
    <SEGTYPE(CONTROLER).NE.'NOUN',
       <-LF<'PARTOF',1"-1.0">>,
    <SUBGOAL2<'PREF'...CONTROLER...ENTRIES<COMPLEMENT,
              'PARTOF'>...SEGTYPE(CONTROLER)...2>,
       <-LF<'PARTOF',1"0.7">>,
    <-LF<'PARTOF',1"-0.3">)
```

Figure 5. Formal specification of a heuristic.

While we do not want to discuss this formalism in any detail, a few points are worth noting. First, each heuristic has indexing information indicating what kind of goal it can solve. This information allows the system to retrieve, easily and efficiently, all the heuristics applying to a given goal. Secondly, each heuristic has formal arguments (HEAD and COMPLEMENT in the case of H2) which will be unified to the effective arguments of the goal it is applied to. All the heuristics designed for the same type of goal have the same arguments.

The body of a heuristic consists of a set of clauses, each of which includes one condition and one action. The clauses are evaluated in sequence until a condition is satisfied; the evaluation of the corresponding action then yields the solution proposed, by the heuristic, for the goal to which it is applied.

Conditions can make use of any function returning a truth value, but can also make explicit calls to "solving operators" for solving subgoals. Thus the call to SUBGOAL2 in Figure 5 starts the process of solving the preference checking subgoal. Such conditions are considered to be satisfied if the resulting certainty factor of the subgoal is positive. This factor is then combined with the certainty factor specified by the action of the clause according to rules of "conditional evidence" explained in Appendix 2.

Up to now, and for the sake of simplicity, we have assumed that a solution to a goal consists simply of one certainty factor, summarizing the final amount of belief or disbelief in the proposition stated by that goal. However, for the kinds of problems we are concerned with, we can also distinguish subsolutions. Knowing which subsolution has provided the resulting certainty factor is an important part of the answer. Thus, for the goal (WITH ⟨EAT⟩⟨FORK⟩), we want to know, if possible, not only if *fork* is a credible *with* complement of *eat*, but also what kind of semantic relation is expressed by this complement (in that case, INSTRUMENT). As another example, when checking if a word satisfies a semantic preference, we might be interested in knowing for which sense of the word the preference is best satisfied.

A solution will then be defined as a list of subsolutions, each subsolution having its own certainty factor:

⟨solution⟩::= (⟨subsolution⟩*)

⟨subsolution⟩::= (⟨answer⟩ . factor).

For the call to CHOOSE given (WITH ⟨EAT⟩⟨FORK⟩), the heuristics of section 4 will yield respectively the following solutions:

H1: ((INSTRUMENT . 0.49))

H2: ((PARTOF . -1))

H4: ((ALL . 0))

The answer provided by H4 makes use of a special subsolution called ALL, which specifies that the corresponding certainty factor (0) applies to all possible subsolutions of the problem. The effect here is to assert that this heuristic has nothing useful to say about this specific goal.

The separate solutions produced by the three heuristics provide "cumulative evidence" which will have to be combined to yield the overall solution to our goal, which will be, in this case:

((INSTRUMENT . 0.49)(OTHERS . 0)(PARTOF . -1))

This solution asserts that the most credible way in which ⟨FORK⟩ can be a *with* complement for ⟨EAT⟩ is by playing an INSTRUMENT role, with a certainty factor of 0.49. The combination of the special ALL subsolution with the other answers has led to the introduction of another special subsolution, called OTHERS, meaning "all subsolutions not mentioned in the current solution." The precise rules used to combine cumulative evidence are stated in Appendix 2.

For the second goal of Figure 4, (WITH ⟨FISH⟩⟨FORK⟩), the final solution, obtained by following the same steps, will be:

((OTHERS . 0)(PARTOF . -0.3)(INSTRUMENT . -1))

Since the plausibility of a solution is defined as the highest certainty factor of its subsolutions, this answer is clearly less favorable then the previous one, and the system concludes that ⟨FORK⟩ should be attached preferably to .

7. Summing up and Looking forward

Conclusions

We have shown that it is possible to consult, automatically, a good machine-readable dictionary as a semantic expert or knowledge base, in order to resolve ambiguities in computational parsing of natural language text. Only prepositional phrase attachment ambiguities have been discussed, but the techniques described here should be extendible to ambiguities that involve all other classes of phrases and clauses. In Appendix 1, we present traces of the disambiguation procedure for the prepositional phrase attachments in sentences (1-3).

All types of information present and available in dictionary entries should eventually be used to help in the disambiguation process: examples, synonyms, and usage notes, along with the definitions themselves. It might soon be possible to build a separate structure of semantic hierarchies, by automatically processing dictionary definitions; but it is too early to impose any rigid form on this sort of knowledge base.

This line of research should benefit all areas of natural language processing — from text analysis to machine translation to data base query — by opening up the prospect of a genuinely broad-coverage approach to meaning and understanding.

Future research

Important areas for future development include the following, some of which constitute interesting difficulties for the disambiguation process:

- Development of new heuristics, and adjustment and improvement of the certainty factors, in order to test the system on a larger scale.
- Analysis of relationships that are signaled by prepositions, conjunctions, and other function words, and of the relational patterns that can be found in dictionary entries;
- Incorporation of other types of dictionary information, from as many dictionaries as possible.
- Addition of information on phrasal verbs (verb-particle pairs, verb preposition pairs), and investigation of how these will help in the parsing process.

Appendix 1: Traces of Processing of Examples

We give here traces of the execution of the three examples discussed in the text. Lines preceded by "***" are comments introduced manually to explain the operations. For reasons of space, we give only the parse trees of the definitions contributing directly to the solution.

EXAMPLE 1.

I ate a fish with a fork.

*** Parse tree for the sentence:

```
DECL   NP      PRON*   "i"
       VERB*   "ate"
       NP      DET     ADJ*    "a"
               NOUN*   "fish"
        ?      PP      PREP    "with"
                       DET     ADJ*    "a"
                       NOUN*   "fork"
```

*** Start of disambiguation process:
*** Evaluation of the first construct:
solving problem: (WITH ⟨FISH⟩ ⟨FORK⟩)
*** H4 has nothing useful to contribute in this case:
solution for heuristic H4: ((ALL 0))
*** Applying H2, the system parses all definitions of
*** "fork," looking for PARTOF patterns. None are

*** found:
solution for heuristic H2: ((PARTOF -0.3))
solution for heuristic H1: ((INSTRUMENT -1))
problem solution: ((OTHERS 0)(PARTOF -0.3)
(INSTRUMENT -1))
*** Evaluation of the second construct:
solving problem: (WITH EAT FORK)
solution for heuristic H4: ((ALL 0))
solution for heuristic H2: ((PARTOF -1))
*** Applying H1, the system looks again in the
*** definitions of "fork", this time looking for
*** INSTRUMENT patterns. The following definition
*** yields three possible preference terms: "take",
*** "pitch," and "dig":

```
NP*     DET      ADJ*     "an"
         ....     "implement"
         PP       PREP     "with"
                  QUANT    AJP      ADJ*     "two"
                           CONJ*    "or"
                           AJP      ADJ*     "more"
                  NOUN*    "prongs"
                  PTPRTCL  VERB*    "used"
         ?        ?        PP       AVP      ADV*     "esp"
                                    PREP     "for"
                                    VP       VERB*    "taking"
                                             PREP     "up"
         ?        ?        ?                PP       PUNC     "("
                                                    PREP     "as in"
                                                    VERB*    "eating"
                                                    PUNC     ")"
                                    CONJ     ","
                                    VP       VERB*    "pitching"
                                    CONJ*    ", or"
                                    VP       VERB*    "digging"
```

*** H3 is applied to each of the terms, attempting to link
*** them to "eat"; a match is found for "take":
solving problem: (PREF ⟨EAT⟩ ⟨TAKE⟩ VERB) solution for heuristic H3: ((TAKE 0.7))
problem solution: ((TAKE 0.7))
solving problem: (PREF ⟨EAT⟩ ⟨PITCH⟩ VERB)
solution for heuristic H3: ((⟨PITCH⟩ -0.7))
problem solution: ((⟨PITCH⟩ -0.7))
solving problem: (PREF ⟨EAT⟩ ⟨DIG⟩ VERB)
solution for heuristic H3: ((⟨DIG⟩ -0.7))
problem solution: ((⟨DIG⟩ -0.7))
solution for heuristic H1: ((INSTRUMENT 0.49))
problem solution: ((INSTRUMENT 0.49)(OTHERS 0)(PARTOF -1))
*** The final result lists all possible constructs,
*** ordered according to decreasing certainty factors.
*** The answer says that "fork" goes more likely with
*** "eat," the underlying relation being
*** INSTRUMENT, with a certainty factor of 0.49:
((WITH ⟨EAT⟩ ⟨FORK⟩)(INSTRUMENT 0.49)(OTHERS 0)(PARTOF -1))
((WITH ⟨FISH⟩ ⟨FORK⟩)(OTHERS 0)(PARTOF -0.3)(INSTRUMENT -1))

EXAMPLE 2.

I ate a fish with bones.

*** Parse tree for the sentence:

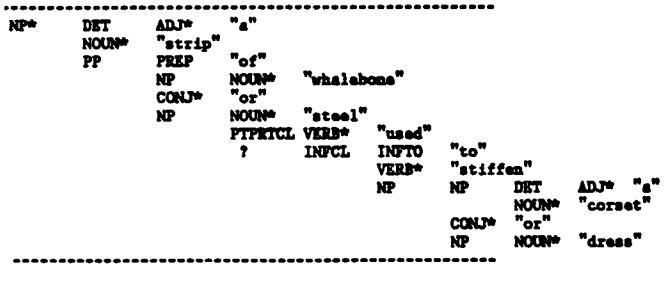
*** Start of the disambiguation process:

```
DECL    NP       PRON*    "i"
        VERB*    "ate"
        NP       DET      ADJ*     "a"
                 NOUN*    "fish"
         ?       PP       PREP     "with"
                          NOUN*    "bones"
```

*** Evaluation of the first construct:
solving problem: (WITH ⟨FISH⟩ ⟨BONE⟩)
solution for heuristic H4: ((ALL 0))
*** Applying H2, the system parses all definitions of
*** "bone," looking for PARTOF patterns. The follow-
*** ing definition is productive:

```
NP*      NOUN*    "one"
         PP       PREP     "of"
                  DET      ADJ*     "the"
                  AJP      ADJ*     "hard"
                  NOUN*    "parts"
          ?       PP       PREP     "of"
                           DET      ADJ*     "the"
                           NOUN*    "skeleton"
          ?       ?        PP       PREP     "of"
                                    DET      ADJ*     "a"
                                    NOUN*    "vertebrate"
```

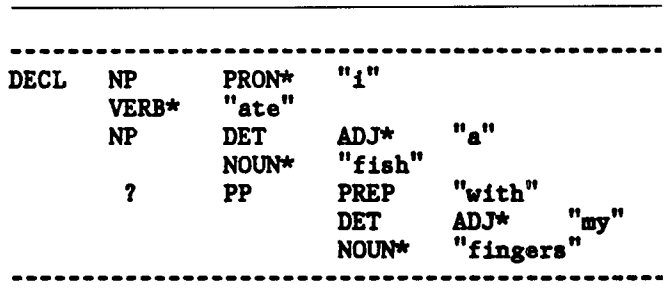
*** The terms found are "skeleton" and "vertebrate";
*** they are checked in turn:
solving problem: (PREF ⟨FISH⟩ ⟨SKELETON⟩ NOUN) solution for heuristic H3: ((⟨SKELETON⟩ -0.7))
problem solution: ((⟨SKELETON⟩ -0.7))
solving problem: (PREF ⟨FISH⟩ ⟨VERTEBRATE⟩ NOUN) solution for heuristic H3: ((⟨VERTEBRATE⟩ 0.7))
problem solution: ((⟨VERTEBRATE⟩ 0.7))
solution for heuristic H2: ((PARTOF 0.49))
solution for heuristic H1: ((INSTRUMENT -1))
problem solution: ((PARTOF 0.49)(OTHERS 0)(INSTRUMENT -1))
*** Evaluation of the second construct:
solving problem: (WITH ⟨EAT⟩ ⟨BONE⟩)
solution for heuristic H4: ((ALL 0))
solution for heuristic H2: ((PARTOF -1))
*** Looking for INSTRUMENT patterns in the defini-
*** tion of "bone", H1 will find one productive defini-
*** tion:
*** The term found is "stiffen". However, this will not
*** provide any link with "eat":
solving problem: (PREF ⟨EAT⟩ ⟨STIFFEN⟩ VERB)
solution for heuristic H3: ((STIFFEN -0.7))
problem solution: ((⟨STIFFEN⟩ -0.7))
solution for heuristic H1: ((INSTRUMENT -0.3))



problem solution: ((OTHERS 0)(INSTRUMENT -0.3)(PARTOF -1)) *** Final result:
((WITH ⟨FISH⟩ ⟨BONE⟩)(PARTOF 0.49)(OTHERS 0)(INSTRUMENT -1))
((WITH ⟨EAT⟩ ⟨BONE⟩)(OTHERS 0)(INSTRUMENT -0.3)(PARTOF -1))

EXAMPLE 3.

I ate a fish with my fingers. *** Parse tree for the sentence:
*** Start of the disambiguation process:



*** Evaluation of the first construct:
solving problem: (WITH ⟨FISH⟩ ⟨FINGER⟩)
*** H4 this time catches the "my" and answers:
solution for heuristic H4: ((PARTOF -1))
*** Applying H2, the system tries to find PARTOF
*** patterns, but there are none:
solution for heuristic H2: ((PARTOF -0.3))
solution for heuristic H1: ((INSTRUMENT -1))
problem solution: ((INSTRUMENT -1)(PARTOF -1))
*** Evaluation of the second construct:
solving problem: (WITH ⟨EAT⟩ ⟨FINGER⟩)
solution for heuristic H4: ((ALL 0))
solution for heuristic H2: ((PARTOF -1))
*** There are no INSTRUMENT patterns either:
solution for heuristic H1: ((INSTRUMENT -0.3))
problem solution: ((OTHERS 0)(INSTRUMENT -0.3)(PARTOF -1))
*** Final result:
((WITH ⟨EAT⟩ ⟨FINGER⟩)(OTHERS 0)(INSTRUMENT -0.3)(PARTOF -1))
((WITH ⟨FISH⟩ ⟨FINGER⟩)(INSTRUMENT -1) (PARTOF -1))

Appendix 2: Operations on Heuristic Solutions

Cumulative evidence is handled by the following operation, which computes a new certainty factor out of two given ones N1 and N2 ("<-"indicates the value returned by the function):

CUMULF
arguments: two certainty factors N1 and N2
result: a new certainty factor
1. if N1.EQUAL.-1 or N2.EQUAL.<-1, -1,
2. if N1.EQUAL.1 or N2.EQUAL.1 , < 1,
3. if (N1+N2).EQUAL.0, <-0,
4. else <- N1+N2-(N1*N2)

Before using this operation, however, we need to know the certainty factor resulting from each subsolution, which is obtained by combining the certainty factor directly assigned to that subsolution and the certainty factor of the special subsolution ALL. This is performed by the operation of **normalizing** a solution, which can be described as follows:

NORMALIZE
argument: a solution
result: this solution, normalized.
1. For each subsolution other than ALL, change its certainty factor to the result obtained by combining it (using CUMULF) with the certainty factor of the ALL subsolution;
2. replace the ALL subsolution, if present, by another special subsolution called OTHERS, with the same certainty factor.

The certainty factor of ALL cannot just be discarded after normalization, because it applies also to all possible subsolutions not mentioned in the current solution; this is taken into account by the special subsolution OTHERS, whose certainty factor applies to all subsolutions not currently mentioned in the solution.

We can now define the operation of combining two solutions:

CUMUL
arguments: two solutions
result: a new solution combining the two given ones as incremental evidence.
1. normalize the two solutions given as arguments;
2. For each subsolution (including OTHERS) occuring in both solutions, define a new certainty factor as the result of **CUMULF** applied to the certainty factors given in the two solutions;
3. For each subsolution occuring in only one solution but such that OTHERS appears in the other solution, define a new certainty factor as the result of **CUMULF** applied to the certainty factors of that subsolution and of OTHERS;
4. For each subsolution not processed in the two preceding steps, keep its certainty factor.

According to these rules, having a subsolution not mentioned in a a solution is equivalent to having it mentioned with a certainty factor of zero.

The processing of conditional evidence (how the

solution given by a heuristic is affected by solutions to subgoals used to execute this heuristic) follows the following rules:

- the **certainty factor of a solution** is defined as the highest certainty factor (after normalization) of the subsolutions of that solution.
- when the solution of a heuristic depends on some subgoal solved by another heuristic, the resulting solution for the heuristic will be obtained by the operation COMBINE defined below:

COMBINE

arguments: two solutions, one for a main problem or goal, and one for a subgoal;

result: a new solution for the goal.

1. Normalize the two argument solutions;
2. Find the highest certainty factor of the solution for the subgoal;
3. Assign to each subsolution in the goal a new certainty factor obtained as the result of COMBINEF applied to the old certainty factor and to the value found in the previous step.

COMBINEF

arguments: two certainty factors N1 and N2,

result: a new certainty factor.

1. If N1.EQUAL.-1 or N2.EQUAL.-1, <-1,
2. <- else N1*N2

REFERENCES

Binot, Jean-Louis. 1985. SABA: vers un systeme portable d'analyse du francais ecrit. Ph.D. dissertation, University of Liege, Liege, Belgium.

Dahlgren, K. and J. McDowell. 1986. Using Commonsense Knowledge to Disambiguate Prepositional Phrase Modifiers. *Proceedings of AAAI-86*, Philadelphia, PA, August 1986.

Funk & Wagnalls editorial staff. 1953. *Standard Handbook of Prepositions, Conjunctions, Relative Pronouns and Adverbs*. Funk & Wagnalls Co., New York.

Heidorn, George E. 1975. Augmented Phrase Structure Grammars. In: Nash-Webber and Schank, eds., *Theoretical Issues in Natural Language Processing*. Association for Computational Linguistics.

Jensen, Karen. 1986. Parsing Strategies in a Broad-coverage Grammar of English. Research Report RC 12147, IBM Thomas J. Watson Research, Yorktown Heights, NY.

Lytinen, Steven L. 1986. Dynamically Combining Syntax and Semantics in Natural Language Processing. *Proceedings of AAAI-86*, Philadelphia, PA, August 1986.

Markowitz, Judith, Thomas Ahlswede, and Martha Evens. 1986. Semantically Significant Patterns in Dictionary Definitions. *Proceedings of the 24th Annual Meeting of the ACL*, Columbia University, June 1986.

Schubert, Lenhart K. 1986. Are There Preference Trade-offs in Attachment Decisions? *Proceedings of AAAI-86*, Philadelphia, PA, August 1986.

Shortliffe, E.H. 1976. *Computer-based Medical Consultation: MYCIN*. Artificial Intelligence Series. Elsevier.

Wilks, Yorick. 1975. An Intelligent Analyser and Understander of English. *Communications of the ACM 18(5)*.