Parsing Spoken Language: a Semantic Caseframe Approach

Philip J. Hayes, Alexander G. Hauptmann, Jaime G. Carbonell, and Masaru Tomita

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213, USA

Abstract

Parsing spoken input introduces serious problems not present in parsing typed natural language. In particular, indeterminacies and inaccuracies of acoustic recognition must be handled in an integral manner. Many techniques for parsing typed natural language do not adapt well to these extra demands. This paper describes an extension of semantic caseframe parsing to restricted-domain spoken input. The semantic caseframe grammar representation is the same as that used for earlier work on robust parsing of typed input. Due to the uncertainty inherent in speech recognition, the caseframe grammar is applied in a quite different way, emphasizing island growing from caseframe headers. This radical change in application is possible due to the high degree of abstraction in the caseframe representation. The approach presented was tested successfully in a preliminary implementation.

1. The Need for Parsing in Speech Understanding

For a computer to understand and respond to a wide range of spoken natural language, it is not sufficient merely to recognize which words were spoken. As in the case of typed natural language input, it is necessary to determine the meaning of the input utterance taken as a whole. The field of natural language processing is devoted to determining the meanings of word sequences typed into a computer. It seems, therefore, natural to apply the techniques already developed in processing typed language to determining the meaning of spoken input.

Unfortunately, it is not possible to apply techniques for parsing typed natural language to spoken input in a straightforward manner. We list some problems below. We assume the existence of a *speech recognizer* that transforms a spoken input into a *word lattice* — a set of hypothesized words that may be present, together with their starting and ending times and the probability of each word being correct. In general, there will be several competing word hypotheses for each point in the input signal. This assumption is somewhat simplistic in that it does not provide any way for a parser to influence the lower levels of speech processing. However, the separation assumption helps to illustrate the following problems in adapting parsing techniques for typed input to spoken input:

- **lexical ambiguity:** More than one word may be produced by the speech recognizer for a given segment of speech. If the ambiguities were simply between different word choices, this could be handled by the natural language processing techniques used for word sense ambiguity (e.g. "bank" may be a place to put money, the side of a river, an action of placing trust, tilting a vehicle sideways, etc.). However, not only can multiple words be hypothesized, but the competing hypotheses can occur at overlapping, adjoining, or separate segments of the input signal, without a consistent set of word boundaries. There is no parallel phenomenon for typed natural language.
- **probability measures:** Speech processing systems typically provide a relative likelihood of the correctness of each word hypothesis. These probabilities or scores are based on criteria such as the quality of the match between speech signal and phonemic dictionary expectations. Since a speech recognition system may hypothesize many words for the same segment of speech, and since these word scores may differ considerably, they are important in limiting the search. However, there is no natural way to make use of such likelihood scores in most natural language processing techniques.
- **unrecognized words:** Because of hurried pronunciation or co-articulation effects, a speech recognizer may completely fail to recognize some words in an utterance. The missed words are usually (though not always) short, unstressed, "function" words rather than longer "content" words. This omission is not handled by standard natural language processing techniques. However, new techniques for processing typed, but grammatically imperfect, input may be adaptable to this purpose since they are also designed to deal with missing words.
- **ungrammatical input:** In addition to the word omissions from imperfect word hypothesization, spoken input tends to contain more real grammatical deficiencies than typed input. Once spoken, words cannot be easily retracted, but typed utterances can be corrected if the user notices the error in time. Thus, fail-soft techniques for recovery from grammatical errors in natural language processing are particularly pertinent when extended to the interpretation of spoken input.

These difficulties argue against the simplistic approach of attaching a speech-recognition module to a traditional natural language analyzer designed for words entered as unambiguous ASCII characters. No matter how good each may be in isolation, the two will not integrate successfully if the latter cannot provide semantic expectations to the former, cannot handle massive lexical ambiguity, or cannot tolerate errors of recognition and grammatical deviation. Moreover, with adequate integration, feedback from a natural language analysis component can substantially improve the performance of a connected speech recognizer. This performance enhancement is badly needed since no present connected speech recognition method comes close to human abilities. And even humans often fail to recognize function words extracted from their surrounding context. The application of linguistic knowledge and semantic expectations through natural language analysis techniques is thus needed to complement acoustic recognition methods by constraining the set of possible (and sensible) interpretations of the words in an input utterance.

2. Problems with Network-based Parsing of Spoken Input

The case for substantial integration of natural language processing with speech recognition is clear. The issue is how to adapt natural language parsing techniques to cope with the special problems of spoken input as described above. Most such adaptation efforts until now have been based on transition network parsing. Essentially, they encode the expectations of the parser in a transition network whose arcs are labelled by syntactic or semantic categories of words or constituents. An input is analyzed by finding a path through the network that corresponds to the sequence of words in the input.

Constituent labels on arcs are associated with their own subnetworks, and traversing the arc in the top-level network is accomplished by traversing the corresponding subnetwork. Typically, transition net parsers operate by traversing the network from left to right in step with the input, exploring subnetworks in a top-down manner as they go. Well known examples of transition-net parsers include ATN [14] parsers (as used in the LUNAR system [15]), the RUS parser [1], and the parser used in LIFER [8]. The HARPY system [9] used an integrated network encoding for linguistic and acoustic information.

A major problem with transition-net parsers for speech recognition lies in the difficulty they have in handling input that does not meet their grammatical expectations. Frequently a word may be missing due to acoustic misrecognition or actual omission. If a network is being explored left to right, finding the correct path through the network would then involve skipping over the arc that corresponded to the missing word. If simple skipping were all that was involved, the problem might well be tractable, but the problem is compounded by the typical multiplicity of possible parses, especially if the word lattice contains many alternative words for the same speech segment. The method used to detect a non-viable parse in the search is inability to follow any arc from the current node — precisely the situation most likely with a missing word. Thus, network parses can no longer use the standard halting criteria for non-productive (constraint violating) searches. A further compounding of the problem arises if the word after the missing word allows a spurious arc to be followed from the network node at which the missing word should have been recognized. In this case, it will generally be very hard to find out where the error really occurred. Other forms of ungrammaticality, either actually spoken or mis-recognition artifacts, result in similar problems. The absence of consistent word boundaries from the acoustic analysis phase complicates things further.

Various methods have been tried to adapt network parsing to these problems, including on-demand insertion of extra arcs (e.g. [13, 12]). Perhaps the most promising modification for speech input is the replacement of left-to-right tracing techniques by center-out techniques that work from words with high certainty according to the acoustic component [16]. However, semantic importance has never been combined with acoustic certainty in selecting these islands. Island growing, attractive in theory, presents serious practical problems for ATN parsers, not the least of which is the requirement of running ATNs from right to left. This method of interpreting the networks, necessary with center-out techniques, fails when tests depend on registers that have not yet been set. No modifications to network-based techniques have been totally successful.

3. Semantic Caseframe Parsing

Our approach is quite different from the transition network approach and is derived from recent work at Carnegie-Mellon University by Carbonell, Hayes, and others [3, 7, 6, 2] on understanding typed, restricted domain natural language, with a particular concentration on handling ill-formed input. The technique that makes it possible to process sensible but potentially imperfect or incomplete utterances is called *semantic caseframe instantiation*. Unlike network-based techniques, caseframe methods enable a parser to anchor its interpretation on the most significant input parts, and to grow its islands of interpretation to the less significant segments. Since the more significant words tend to be longer and therefore more likely to be recognized reliably, the islands of significance are correlated with islands of certainty. In the process, semantic and syntactic expectations generated from the more meaningful parts of the input can be used to discriminate and hypothesize the meaning of troublesome segments.

The essential difference between caseframe and transition network techniques is the level of encoding of the syntactic and semantic information they both use. Caseframe techniques encode the information at a more abstract level and thus are able to interpret it in multiple ways. Network techniques "compile" the information into networks at a much lower and more rigid level, and thus do not have nearly as much freedom in interpreting the same knowledge in multiple ways. As we will show, the ability to apply syntactic and semantic information in an interpretive way is the key to the successful integration of speech and natural language processing.

The central notion behind a caseframe is that of a head concept modified by a set of related concepts or *cases*, bearing well-defined semantic relations to the head concept. The original linguistic concept of a caseframe as first described by Fillmore [4], relied on a small set of universally applicable cases. The recent work at CMU adapts this idea to restricted domain situations by allowing specialized cases for each concept related to a head concept. Consider, for instance, the caseframe shown in Figure 1.

```
#S(ED
  Name ForwardAction
  Type verb
  SemanticCases (
    #S(SC
      Name    Agent                    ;the sender
      InstanceOf (MailAdrDesc)
      SyntaxCase (Subject))
    #S(SC
      Name    MsgObj                   ;a message
      InstanceOf (MsgObjDesc)
      SyntaxCase (DirectObject))
    #S(SC
      Name    MsgRecipientObj          ;the receiver
      InstanceOf (MailAdrDesc)
      SyntaxCase (IndirectObject PrepO)
      CaseMarker (to))
    #S(SC
      Name    CCRecipientObj           ;the CarbonCopy
      InstanceOf (MailAdrDesc)         ;receiver
      SyntaxCase (PrepO)
      CaseMarker (ccing copying)))
    RequiredSC (MsgObj MsgRecipientObj Agent))
  HeadForms (forward resend))
```

Figure 1: Caseframe for *forward*

Figure 1 defines the *forward* action of an electronic mail system. The notation is that of the caseframe speech parser described later. Without going into notational details, the caseframe is identified as a verb or clausal caseframe corresponding to the verbs (HeadForms) "forward" or "resend". It also has four cases: Agent (the person doing the sending), MsgObj (the message being forwarded), MsgRecipientObj (the person the message is being forwarded to), and CCRecipientObj (the people who get a copy of the forwarded message). The MsgObj case must be filled (InstanceOf) by a MsgObjDesc (defined by another caseframe, see below), and the other cases must be filled by a MailAdrDesc (the caseframe representing a person or "mail address"). All the cases are required, except CCRecipientObj, which is optional. In addition, to this purely semantic information, the caseframe contains some syntactic information: the Agent case is manifested as the syntactic subject; MsgObj as the direct object; MsgRecipientObj as either the indirect object or as the object (PrepO) of a prepositional phrase, whose preposition (CaseMarker) is "to"; CCRecipientObj as a prepositional

phrase with "prepositions" either ccing or copying.

```
#S(ED
  Name MsgObjDesc
  Type Noun
  SemanticCases (
    #S(SC
      Name    Descriptors
      Pattern (new recent old unexamined examined)
      SyntaxCase (prenominal))
    #S(SC
      Name    Determiners
      Pattern (the this that any a every)
      SyntaxCase (prenominal))
    #S(SC
      Name    MsgOriginObj       ; where the mail
      InstanceOf (MailAdrDesc) ; came from
      CaseMarker (from)
      SyntaxCase (Prep0))
    #S(SC
      Name    TimeObj
      InstanceOf (HourDesc MonthDesc DayDesc)
      CaseMarker (from before after since on at)
      SyntaxCase (Prep0)))
  HeadForms (message mail))
```

Figure 2: Caseframe for ***message***

In addition to actions, we also use caseframes to describe objects Figure 2 shows a nominal caseframe for the message object of our electronic mail system. This has the same form as the verb caseframe, except that its HeadForms correspond to the head nouns of a noun phrase describing an electronic mail message. In addition, the Descriptors case has a new SyntaxCase, prenominal, which implies that the elements of Pattern (new, recent, etc.) may appear in the adjective position in this caseframe.

With a suitable caseframe for MailAdrDesc and knowledge of what things like clause, noun phrase, direct object, adjective position, etc. mean, the above caseframes clearly contain enough information to produce analyses of sentences like:

Forward to Jones at CMUA the messages from Smith.
Did Brown resend any new messages to Green at BBN?
What mail did Jones forward to Smith?
Brown is forwarding the recent messages to Green.

The central question is how to combine the information in the caseframe definitions with syntactic knowledge and thus analyze the sentences into a set of caseframe instances.

The approach taken in earlier caseframe work at CMU has been to embed the syntactic knowledge in the parser code and let the parser interpret the caseframes using that knowledge. E.g. the algorithms in [3] use semantic caseframes and focus on prepositions as casemarkers as well as the order of subject, verb, indirect object and direct object for parsing. Unfortunately, prepositions tend to be small function words that are often poorly enunciated and recognized. Therefore we have adopted the same general approach for our speech parsing work, but modified the parsing algorithms. The same caseframes are used, but with a somewhat different interpretation process.

The ability to apply multiple recognition methods is a central advantage of caseframe parsing. Since the restricted-domain language description embodied in the caseframes is at such a high level of abstraction, we are free to interpret it in a way appropriate to the particular situation. The caseframes tell us ***what*** components to look for and constrains ***where*** we can look for them. But exactly ***how*** we look for them is adaptable so that it can be driven by the most reliable information we have.

4. Applying caseframes to speech input

We can summarize the previous two sections as follows:

- caseframes of the kind we have described contain the right amount of information at the right level of abstraction to parse restricted-domain spoken input;
- the algorithms that have been developed for using such caseframes in parsing typed natural language input are unsuitable for spoken input because the algorithms rely on the presence of small function words that are recognized at best unreliably by word hypothesizers.

The trial implementation of our approach applies caseframes to the input, but does it in a novel way by:

1. examining the lattice of words hypothesized by the speech recognizer for those that correspond to caseframe headers
2. combining all the caseframes corresponding to the words found in all semantically and syntactically plausible ways
3. for each caseframe combination thus formed, attempting to account for the gaps between the caseframe header words that were involved in its formation by parsing words from the gaps against empty semantic and syntactic roles in the caseframe combination
4. selecting as the final parse those caseframe instances that best account for the input, based on how much input they cover and the acoustic scores of the words in that parse.

This multi-stage approach avoids the problems of the caseframe parsing algorithms for typed input by anchoring the parse on caseframe headers. Caseframe headers are verbs (for clausal caseframes) and nouns (for nominal caseframes). These are content bearing words that tend to be stressed in speech and are often multi-syllabic. This improves their chances of recognition above that of short, unstressed function words. The anchor points are thus correlated to the most acoustically certain words.

The idea of forming one or more parses at a skeleton level and instantiating the one (or ones) that satisfy all constraints down to the lexical level is akin to the ABSTRIPS [10] and NOAH [11] planners that first established a general plan and later worked in all the detail called for in the situation. That way, the parser does not waste time in hypothesizing local details that cannot possibly fit into a global parse.

An additional advantage associated with working from caseframe headers is that the resulting caseframe combinations form a ready-made semantic interpretation of the input. The interpretation is typically incomplete until it is filled out in the subsequent gap-filling stage. However, if the recognition of some or all of the remaining words is so poor that the semantic interpretation is never fully completed; then the parser still has something to report. Depending on the application domain, a skeleton interpretation could be sufficient for the application, or would at least form the basis of a focussed request for confirmation or clarification to the user [5].

In the remainder of this section, we examine in more detail our current implementation of the approach outlined above, starting first with a description of the word lattice that drives our caseframe-based parser for spoken input. This parser operates in the context of a complete speech understanding system that handles speaker independent continuous speech with a 200 word vocabulary in an electronic mail domain.

4.1. The word lattice

The input to our caseframe speech parser can be viewed as a two-dimensional lattice of words. Each word has a begin time, an end time, and a likelihood score. The begin/end times state where the word was detected in the utterance. The score indicates how certain we are that the word is correct, based on acoustic-phonetic information. In the sample lattice below, the horizontal dimension is time, and the vertical dimension corresponds to certainty of recognition of individual words by the speech recognizer generating the lattice. This word lattice was constructed by hand for demonstration purposes.

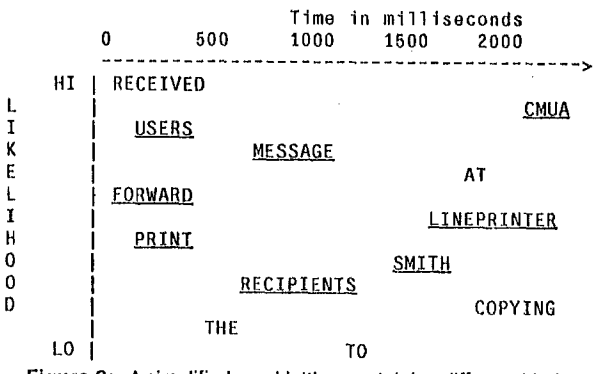

Figure 3: A simplified word lattice containing different kinds of words. Header words are underlined

4.2. Header combination

To start its processing, the parser selects from the word lattice all header words above a recognition likelihood threshold. These headers correspond to caseframes, but only some combinations of the hypothesized caseframes are possible in the domain. To calculate the legal caseframe combinations, a set of phrase structure rules were derived that apply at the frame level (rather than at the more detailed word level).

To make matters more concrete, let us refer to the sample lattice above. In this lattice, the underlined header words would be combined to form the nuclei of sentences like: *"Forward message Smith CMUA"* and *"Print message lineprinter."* Caseframes can combine in this way if one is of the right type (as defined by the InstanceOf attribute for the case) to fill a case of another. When combining caseframes associated with header words, the parser also uses knowledge about word order to limit the possible combinations. In our example, the *forward* caseframe (as defined in Figure 1) has a slot for a MsgObjDesc as a DirectObject. The order restrictions built into the parser only allow for the direct object after the verb. The *message* caseframe (Figure 2) fulfills these requirements. It is a MsgObjDesc, whose HeadForm "message" occurs after the *forward* caseframe HeadForm "forward" in the lattice. Thus the two can be combined, as long as the constraint of the required MsgRecipientObj can be satisfied (by "Smith").

Each time a valid sequence of headers is found, it is given an overall likelihood score and merged with the previous ones. At the end of the header combination phase, we have a list of ordered partial phrases, containing all the legal sequences of header words that can be found in the word lattice. Each partial phrase is represented as a set of nested caseframe instances. For instance, three combinations would be formed from the header words:

Forward message Smith CMUA

and these would have the nesting structure:

```
[ForwardAction
  HeadForm FORWARD
  MsgObj [MsgObjDesc
            HeadForm MESSAGE]
  MsgRecipientObj [MailAdrDesc
                     HeadForm SMITH
                     Host  [LocationDesc
                              HeadForm CMUA]]]

[ForwardAction
  HeadForm FORWARD
  MsgObj [MsgObjDesc
            HeadForm MESSAGE]
  CCRecipientObj [MailAdrDesc
                     HeadForm SMITH
                     Host  [LocationDesc
                              HeadForm CMUA]]]

[ForwardAction
  HeadForm FORWARD
  MsgObj [MsgObjDesc
            HeadForm MESSAGE
            MsgOriginObj
               [MailAdrDesc
                  HeadForm SMITH
                  Host  [LocationDesc
                           HeadForm CMUA]]]]
```

where square brackets indicate caseframe instances and the nesting is conveyed by textual inclusion.

A routine to check word junctures is used during the header combination phase. Whenever two header words are combined for a partial phrase, the juncture between these words is checked to ascertain whether they overlap (indicating an illegal combination), abut, or have a gap between them (indicating significant intervening speech events). This check also enables the parser to deal efficiently with co-articulated phonemes as in *"some messages"*. These phonemes are merged in pronunciation, resulting in a pair of overlapping but valid word candidates. These word juncture checks comprise a top-down feedback mechanism to improve the speech recognition.

4.3. Casemarker connection

Once caseframe combinations have been formed, the next step is to fill in the gaps between the words of the corresponding partial phrase. We take each combination in turn, starting with the one with maximal-likelihood. The caseframe speech parser first tries to fill in casemarkers, which are usually prepositions.

Let us continue our example with the first header combination formed from the phrase *"Forward message Smith CMUA"*. For this phrase, casemarkers may appear before the prepositionally marked cases *"Smith"* and *"CMUA"*. The requirement that the casemarkers must appear between the header words of the containing and contained caseframes is a strong constraint on the possible locations of the casemarkers. There are generally strong limitations on what words could possibly serve as markers for these cases. In our example, using the caseframe definitions of the previous section, the parser would thus try to verify one of the words *"to"*, *"from"*, *"ccing"* or *"copying"* between *"message"* and *"Smith"* and one of the words *"on"* or *"at"* between *"Smith"* and *"CMUA"*.

Whenever a set of words are predicted by the parser in a given segment, a word verification module is called. This module has knowledge of the complete word lattice. A word that matches the prediction is sought from the lattice in the specified gap. In addition,

the acoustic-phonetic data is consulted to give an indication whether the word is a perfect fit for the gap, a left or right anchored fit, or if there are intervening significant speech events on the left or right. This information allows the parser to determine how much input has been accounted for by a given partial phrase hypothesis.

Every successfully verified casemarker causes the parser to spawn another partial phrase hypothesis. The word could be a spuriously hypothesized word, i.e. one that was "recognized" even though it was never spoken (also known as a false alarm). Therefore we leave the old partial phrase without the casemarker in the ordered list of partial phrases and merge a new partial phrase into the list. The new partial phrase is a copy of the old one, with the casemarker also filled in. A new likelihood score is computed for this phrase.

The score for a partial phrase is currently computed as the sum of the time normalized probabilities of each word divided by the time of the total utterance. Thus the probability of each word is multiplied by the duration of the word, summed over all words and divided by the duration of the utterance. This favors longer partial phrases over shorter ones. However, even extremely low scoring long phrase candidates are favored over well scoring shorter phrases. We are currently also exploring other alternative scoring procedures for

4.3. Casemarker connection

Once caseframe combinations have been formed, the next step is to fill in the gaps between the words of the corresponding partial phrase. We take each combination in turn, starting with the one with maximal-likelihood. The caseframe speech parser first tries to fill in casemarkers, which are usually prepositions.

Let us continue our example with the first header combination formed from the phrase *"Forward message Smith CMUA"*. For this phrase, casemarkers may appear before the prepositionally marked cases *"Smith"* and *"CMUA"*. The requirement that the casemarkers must appear between the header words of the containing and contained caseframes is a strong constraint on the possible locations of the casemarkers. There are generally strong limitations on what words could possibly serve as markers for these cases. In our example, using the caseframe definitions of the previous section, the parser would thus try to verify one of the words *"to"*, *"from"*, *"ccing"* or *"copying"* between *"message"* and *"Smith"* and one of the words *"on"* or *"at"* between *"Smith"* and *"CMUA"*.

Whenever a set of words are predicted by the parser in a given segment, a word verification module is called. This module has knowledge of the complete word lattice. A word that matches the prediction is sought from the lattice in the specified gap. In addition, the acoustic-phonetic data is consulted to give an indication whether the word is a perfect fit for the gap, a left or right anchored fit, or if there are intervening significant speech events on the left or right. This information allows the parser to determine how much input has been accounted for by a given partial phrase hypothesis.

Every successfully verified casemarker causes the parser to spawn another partial phrase hypothesis. The word could be a spuriously hypothesized word, i.e. one that was "recognized" even though it was never spoken (also known as a false alarm). Therefore we leave the old partial phrase without the casemarker in the ordered list of partial phrases and merge a new partial phrase into the list. The new partial phrase is a copy of the old one, with the casemarker also filled in. A new likelihood score is computed for this phrase.

The score for a partial phrase is currently computed as the sum of the time normalized probabilities of each word divided by the time of the total utterance. Thus the probability of each word is multiplied by the duration of the word, summed over all words and divided by the duration of the utterance. This favors longer partial phrases over shorter ones. However, even extremely low scoring long phrase candidates are favored over well scoring shorter phrases. We are currently also exploring other alternative scoring procedures for partial phrases. These methods will recognize the tradeoff between long, low scoring utterances that seem to account for all the input and short phrase hypotheses with excellent scores that leave gaps in the utterance unaccounted for. An ideal scoring function would also use semantic and syntactic wellformedness as criteria.

Sometimes, none of the case markers being verified are found. This may mean that:

- the speech recognizer failed to detect the marker. Unvoiced co-articulated monosyllabic words (such as prepositions) often go undetected;
- or, the most-likely parse at the case-header level was indeed incorrect, and a lower likelihood parse should be explored to see if it is more consistent with the acoustic data.

At present only the second choice is considered, but we are exploring the possibility of an enhanced verifier to re-invoke the lower level processes (acoustic analysis or word hypothesizer modules) with strong expectations (one or two words) at a prespecified window in the input. We hope that such a process can detect words missed in a more cursory general scan — and thus use semantic and syntactic expectations to drive the recognition of the most difficult segments of the input. If the verifier were to return with a recognized case marker, but too low a likelihood, the overall likelihood value of the next parse could make it the preferred one.

4.4. Prenominal filling

The next phase fills in the prenominal sections of the partial phrases. The parser looks for prenominals in the following order:

*Predeterminer Determiner Ordinal Cardinal Adjective**

A lexicon associates each potential prenominal word with the correct type. Thus we first look for all possible predeterminers (e.g. 'all') within the available gap before the corresponding header word. Again the successful verification of such a prediction spawns a new partial phrase, just as described for casemarkers. The old partial phrase remains in the list as a precaution against false alarms. It should be noted that remaining old phrases accounting for less input receive a lower global likelihood value because unaccounted for input is penalized.

Then determiners are examined. In our example, the determiner "the" will successfully be found to modify the message caseframe. The other prenominal types are filled in the same way. Post-nominal modifiers (i.e., prepositional phrases) are parsed by the caseframe instantiation method above, as nominal and sentential caseframes are treated in much the same way.

4.5. Extending coverage to simple questions

Although we have not made completeness of syntactic coverage a focus in this work (see next section), we made some simple extensions to gain some idea of the difficulty in syntactic extension. In particular, we extended the system to deal with simple interrogatives as well as imperatives and declaratives. No changes to the caseframes themselves were necessary, just to the parsing algorithm. We introduced a separate stage in processing to look

exclusively for question words. These words may be the standard wh-words (who, what, when, ...) or sentence-initial auxiliary verbs to indicate a yes/no question (do, does, is, will, ...).

The word order rules in the header combination phase also required extension. These rules now have to allow fronted cases

What messages did Smith send

and questions where the HeadForm of the case is collapsed into a question word

Who sent this message

. Finally, we added a new module to fill auxiliary verbs in the correct locations. It operates just like the casemarker connection module and will not be described further here. By providing the parser with constraints governing the agreement of subject/verb, of auxiliary verb/main verb, and of prenominal/noun, the number of plausible alternatives is kept low.

5. Summary and Future Directions

We have explored an approach to parsing restricted-domain speech based on semantic caseframes. The approach was shown capable of dealing with the uncertainties and ambiguities of speech and the common ungrammaticalities. We argued that a caseframe approach was better suited to these problems than more traditional network-based approaches. This suitability was attributed to the high degree of abstraction with which caseframes represent their linguistic information, and the corresponding flexibility in interpretation this allows. A simple implementation using this approach was described with a worked example.

We envision continued development of our system and enhancements to our approach in several directions:

- Our current approach relies too heavily on finding caseframe header words. While most are multi-syllable and easily recognizable at the acoustic level, many (e.g. 'send') are not. We are looking at ways to drive the recognition from the most reliably recognized words, whether they correspond to caseframe headers or not.
- Most of the syntactic knowledge used by our current system is embedded in the code. While this makes for efficient and robust recognition, it poses obvious problems for syntactic extensibility and maintainability. We are looking at ways of separating out the syntactic knowledge, while retaining the power and flexibility inherent in specifying a restricted-domain language through caseframes, rather than (say) rewrite rules.
- The nature of the interpretation performed by the present system causes it to operate at large multiples of real-time. We are looking at methods of compiling the caseframe grammar into more efficient recognition systems, with the eventual goal of real-time operation, while retaining our current flexibility and robustness.

6. Acknowledgements

The authors wish to express their debt to all the other members of the speech-natural language project at CMU. J. Siegel programmed large parts of the system with L. Baumeister and H. Saito. R. Reddy and R. Stern coordinated the natural language and speech parts of the system. A. Rudnicky and others provided the acoustic data analysis in the word-lattice form.

This research was sponsored by Defense Advanced Research Projects Agency Contract N00039-85-C-0163. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

References

1. Bobrow, R. J. The RUS System. BBN Report 3878, Bolt, Beranek, and Newman, 1978.

2. Carbonell, J. G. and Hayes, P. J. "Recovery Strategies for Parsing Extragrammatical Language". *Computational Linguistics 10* (1984).

3. Carbonell, J. G. and Hayes, P. J. Dynamic Strategy Selection in Flexible Parsing. Proc. of 19th Annual Meeting of the Assoc. for Comput. Ling., Stanford University, June, 1981, pp. 143-147.

4. Fillmore, C. The Case for Case. In *Universals in Linguistic Theory*, Bach and Harms, Ed., Holt, Rinehart, and Winston, New York, 1968, pp. 1-90.

5. Hayes P. J. A Construction Specific Approach to Focused Interaction in Flexible Parsing. Proc. of 19th Annual Meeting of the Assoc. for Comput. Ling., Stanford University, June, 1981, pp. 149-152.

6. Hayes, P. J. Entity-Oriented Parsing. COLING84, Stanford University, July, 1984.

7. Hayes, P. J. and Carbonell, J. G. Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update. Proc. Seventh Int. Jt. Conf. on Artificial Intelligence, Univ. of British Columbia, Vancouver, August, 1981, pp. 432-439.

8. Hendrix, G. G. Human Engineering for Applied Natural Language Processing. Proc. Fifth Int. Jt. Conf. on Artificial Intelligence, MIT, 1977, pp. 183-191.

9. Lowerre, B. The HARPY Speech Recognition System. Computer Science Department, Carnegie-Mellon University, April, 1976.

10. Sacerdoti, E. D. "Planning in a Hierarchy of Abstraction Spaces". *Artificial Intelligence 5*, 2 (1974), 115-135.

11. Sacerdoti, E. D.. *A Structure for Plans and Behavior.* Amsterdam: North-Holland, 1977.

12. Weischedel, R. M. and Sondheimer, N. K. "Meta-Rules as a Basis for Processing Ill-formed Input". *Computational Linguistics 10* (1984).

13. Weischedel, R. M. and Black, J. "Responding to Potentially Unparseable Sentences". *American Journal of Computational Linguistics 6* (1980), 97-109.

14. Woods, W. A. "Transition Network Grammars for Natural Language Analysis". *Comm. ACM 13*, 10 (Oct. 1970), 591-606.

15. Woods, W. A., Kaplan, R. M., and Nash-Webber, B. The Lunar Sciences Language System: Final Report. 2378, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1972.

16. Woods, W. A., Bates, M., Brown, G., Bruce, B., Cook, C., Klovstad, J., Makhoul, J., Nash-Webber, B., Schwartz, R., Wolf, J., and Zue, V. Speech Understanding Systems - Final Technical Report. 3438, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1976.