音声言語情報処理 2－4
（1994．7．7）

The Application of a Lightweight Parser for Speech Understanding

Nigel WARD

Department of Mechano-Informatics,
Faculty of Engineering,
University of Tokyo

nigel@sanpo.t.u-tokyo.ac.jp

Fully integrated speech understanding has been hard to acheive, in part because of the difficulty of building a syntactic mechanism able to support the computation of feedback from the semantic levels to the word-spotting level. This paper describes a syntax mechanism suitable for this. It embodies a new view of the role of syntax, and exploits three innovations: "construction hypotheses", "participatory profiles", and "timeline-based computation". Experiments show that use of this parser does allow the use of semantic feedback via syntax.

音声理解のための軽量パーザーの実験

N. ワード
東京大学工学部機械情報工学科

統合的な音声理解システムを作成しようとすると、パーザーがネックとなる。本稿では文法構造仮説、関与プロフィール、タイムラインという三つの要素に基いているパーザーを提案し、これを使って概念層から単語認識層へのフィードバックの実現を述べる。

1 Background

Since purely bottom-up speech recognition is generally impossible, it is important to exploit "higher-level knowledge". Thus there is currently much interest in "tightly-coupled" architectures for speech understanding (Seneff 1992; Jurafsky *et al.* 1994), that is, architectures where semantic considerations are fully exploited to aid low-level recognition. This is of course a step towards the long-held goal of building fully integrated speech systems.

The hope is that building such systems will allow more accurate and robust understanding. When dealing with noisy inputs or spontaneous speech, this will be even more necessary.

What we want, therefore, is a system that can, given a partially recognized input, infer something about the meaning, and then use that partial understanding to infer what other words are likely to be in the input. What makes this hard is that this feedback should indicate not only what words are likely, but also where they are likely to appear. This of course requires reasoning about word position and word order; therefore syntactic knowledge must be employed.

This has been the bottleneck. While there are many architectures which support the tightly-coupled integration of knowledge from multiple "knowledge sources", blackboard architectures, for example, there are currently no parser designs which support this integration.

(Of course, there do already exist some integrated speech systems, but in these syntactic and semantic processing is generally handled by a single module. One example of this is parsers which apply some approximate semantic knowledge, for example, "*agents* tend to be *animate*". Another example is "meaning-driven parsers", which apply some fragmentary syntactic knowledge from the semantic engine. Another is semantic grammars, that is, grammars in which semantic constraints are encoded in the grammar. Yet another is the integration of knowledge for semantic disambiguation into the HMM formalism. All these approaches do allow some conceptual knowledge to be applied early, but only certain types of conceptual knowledge. Looking forward to the day when more general semantic interpreters will be available, it seems better to keep meaning and syntax independent. When this is done, the parsing problem becomes central.)

I have been developing a syntactic mechanism suitable for fully integrated speech understanding. I call it a "lightweight parser": "parser" because it applies syntactic knowledge (although, unlike most parsers, it does not output a parse tree), and "lightweight" because invoking it is low-overhead (compared with traditional parsers, which typically run as a separate process (either invoked as a subroutine, or running as a co-routine accepting messages)). It is also "lightweight" in the sense that it requires a minimum of computation and a minimum of working memory.

This paper describes this parser and its use. Section 2 discusses the role of syntax in speech understanding problem, and section 3 presents a lightweight parser. Sections 4 and 5 illustrate the operation of the system, focusing on how hypotheses are scored and on how the parser allows semantic feedback. Section 6 points out some directions for future work.

2 The Role of the Parser

This section considers how a parser should be integrated with the other tasks in speech understanding. Further discussion appears in (Ward 1993).

Traditionally, the parser's input is one or a few strings of words, namely, "sentence hypothesis". The problem with this is that it requires a separate process to prune out word hypotheses or organize them into sentence hypotheses. Doing so requires a "language model", and involves the premature resolution of uncertainty, before higher-level knowledge is fully applied. Therefore a parser should, instead, work directly from the lattice of word hypotheses [1].

Traditionally, the parser's output is one (or a few) full syntactic structures, each spanning the entire input. These are then passed on the semantic component, for transformation to logical form and subsequent use in reasoning. Instead, a parser's output should be no more than a "set of clues" to a semantic interpretation[2]. For any input there will be many such clues, generally inconsistent, with as-

[1] This allows more effective integration for feedback too. For example, even probabilistic parsers typically can only rescore the n-best sentence hypotheses, not individual word hypotheses. That is, most do not actually apply feedback in a way usable for guiding the recognition search (but see Jurafsky *et al* (1994).)

[2] For many natural language tasks, the problem is primarily one of recognizing 'frames' or 'situations' of interest (Martin 1989) and identifying relationships among concepts (Bates *et al.* 1993). Building a semantic interpreter for such tasks may, I suspect, be easier, not harder, if it can work directly from clues, rather than having to deal with full-sentence-sized logical forms.

sociated scores [3].

Thus we can view speech understanding as being basically the flow of information through a network of lexical, syntactic, and conceptual hypotheses. "Parsing" is simply the part of this computation which involves syntactic hypotheses. In particular, given word hypotheses, the parser produces syntactic hypotheses and conceptual hypotheses (clues). The major part of its work is scoring these various hypotheses; that is, computing the evidence for one hypothesis based on its supporting hypotheses. This approach allows fine-grained communication between all components of the system; this communication takes the form of evidence for various hypotheses. This is true for feedback as well; if some reasoning using domain knowledge leads to the rescoring of some conceptual hypothesis, the implications for syntactic and lexical hypotheses are easily computed; that is, their scores can be updated easily by the parser. Thus the parser is truly lightweight — it is available throughout the understanding process, with no overhead.

3 The Design of the Parser

Having now characterized abstractly what a lightweight parser should do, it is time to discuss how to build such a parser.

3.1 Construction Hypotheses

The parser will of course need to consider various syntactic hypotheses. It is important that they be simple and "independent". In other words, the representation of the parser's current "theory" as to the structure of the input should be factored into independent syntactic hypotheses. This is important for two reasons. First, it allows painless parallel consideration of large numbers of raw word hypotheses. (The problem this avoids is that of a combinatorial explosion of the number of parses, which is inevitable if a parser creates complex syntactic hypotheses by coordinating, unifying, assembling, or linking the elements of a syntactic interpretation into trees or other structures.) Second, it allows each syntactic hypothesis to be directly and individually related to the clues (that is, the elements of a conceptual interpretation).

In my system, the syntactic hypotheses used are "construction hypotheses" (Ward 1993).

The key idea is that it is possible to represent syntactic knowledge as an inventory of constructions, analogous to the representation of lexical knowledge as an inventory of words. Each construction is a pairing of form and meaning. The form is a sequence of constituents. Examples of constructions are the Subject-Predicate Construction, the Transitive Construction, the Adjective-Noun Construction, and the Passive Construction. It seems that constructions are a representational mechanism adequate for writing complete grammars; and doing so is the enterprise of "Construction Grammar" (Fillmore 1988).

Adopting this view of grammar allows a parser to use syntactic hypotheses of the form "constituent X of construction Y was present over time span Z". Such "construction hypotheses" have the advantages of being simple, being suited to consideration in parallel, being easily scorable based on word hypothesis scores, and relating directly to semantics.

A construction hypothesis can be spawned when there is a good match between the constituents of the construction and the hypothesized words in some time range.

A construction hypothesis which spans a certain time range can be used for interpreting that part of the input. To give just one example, suppose that: A. there is a hypothesized occurrence of the Subject-Predicate Construction for which the first constituent spans the time span from the 10th to the 22nd frame, and B. an occurrence of the word *"John"* is hypothesized in the time span from the 11th to the 19th frame. From this, since the time spans overlap, there is evidence for *john* being the subject. This then is used to compute the clues to pass on to the semantic interpreter.

Conversely, for feedback, semantic rescoring of such a clue directly causes rescoring of the associated construction hypothesis (or hypotheses).

In that it makes syntactic hypotheses parallel, scored, and independent, this approach combines the best points from chart parsing, probabilistic parsing, and partial parsing (Ward 1993).

[3] By adopting this parser/interpreter interface there is no special difficulty with ungrammatical, fragmentary, or noisy inputs; the parser can simply extract as many clues as possible, in the normal way. This avoids the need for special processes (fallback algorithms) to handle "ill-formed inputs". That is, this approach treats complex, incomplete inputs as the normal case. This contrasts with approaches which assume that the normal input is an unambiguous string of words with only one interpretation.

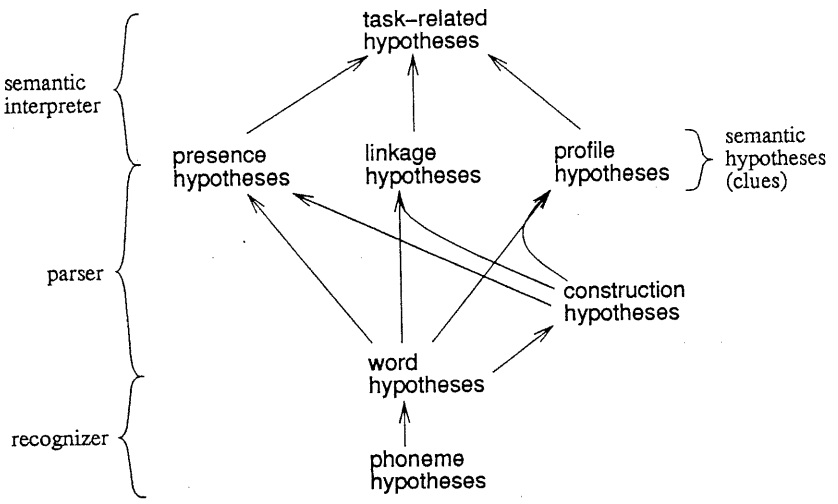

Figure 1: Hypothesis types and their relations. The arrows represent forward evidential relations. Feedback pathways are not shown.

3.2 Participatory Profiles for Meaning Representing

For ease of feedback, it helps if the semantic interpreter and syntactic mechanism use the same representation mechanisms, as much as possible.

The system accordingly uses "participatory profiles" (Ward 1992; Ward 1994) instead of Fillmorean deep cases. In brief, the idea is to decompose cases into vectors over "case features". For example, the semantics of the first constituent of the Subject-Predicate Construction is not "*agent*" but the participatory profile *(.6 topic), (.4 volitional), (.3 active), (.5 responsible), (-.2 affected)* .... The evidence is smoothed; for example, (.6 topic) also counts as (weaker) evidence for *(.5 topic)*.

This obviates the need for a case-slot mapping. Moreover, it makes it easy for many construction hypotheses to independently contribute clues to an interpretation; these clues can be combined numerically.

Thus, I propose that the clues (that is, the simple conceptual hypotheses which the parser scores) are of three types: 1. presence information, e.g., "this input involves *john*", "this involves a *beneficial action*", 2. linkage-information, e.g. "in this input *john* is related to *kiss*", and 3. relational information, e.g. "*john* is *active* in this input". Note that, this inventory decomposes traditional case relations, such as "*john* is the *agent* of *kiss*", into one clue of type 2 and several clues of type 3.

Figure 1 summarizes these points.

3.3 Timeline-Based Computation

As mentioned above, the system relies on "overlapping time spans" for computing hypotheses scores. This is done primarily for the sake of simplicity. The problem it avoids is that of binding words to the constituents of constructions.

The technique used in this system can be called "computing with timelines". For example, if there is the hypothesis that "the part of the input from frame11 thru frame18 corresponds to direct-object", then there is evidence for *(affected .8)*, and this evidence is stored on positions 11 thru 18 of the *(affected .8)* timelines. If there is also the hypothesis that "the word *"John"* appeared from frame10 thru frame 16", then, by using the information on the timeline, the evidence regarding the degree of affectedness of *John* can be easily computed.

Evidence from many construction hypotheses (etc) is summed onto each timeline. As there is only a small number of timelines, much fewer than the number of hypotheses of various types, this technique is relatively fast.

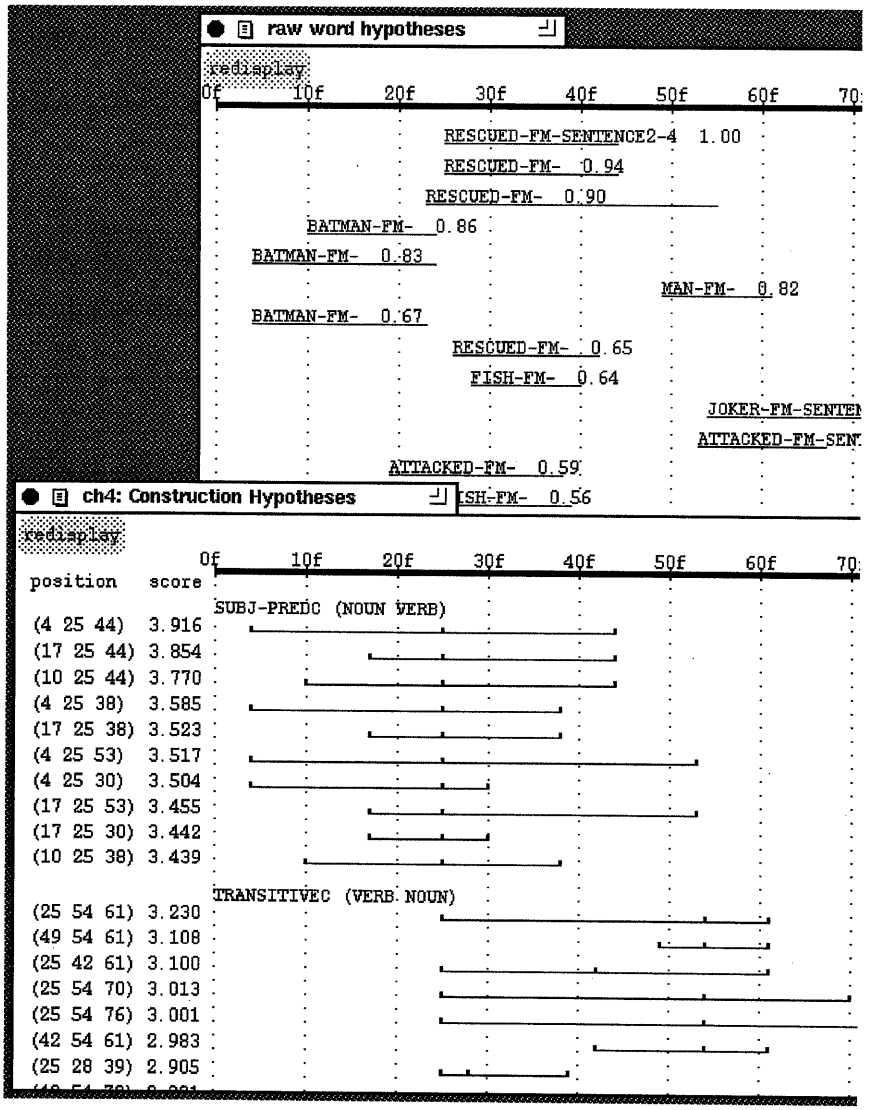

Figure 2: The output of the word spotter. The numbers are the scores of the word hypotheses and the horizontal lines show their time spans. The dotted lines mark off 200 ms intervals.

Figure 3: Some construction hypotheses. The horizontal line segments show the time spans of the constituents. Note the independence (indeed, inconsistency) of these construction hypotheses. For example, according to the best ranked Subject-Predicate hypotheses the verb ends at frame44, but according to the best Transitive hypothesis, the verb ends at frame54.

4 The System

I have implemented a speech understanding system, consisting of about 1000 lines of C and 4000 lines of Lisp. It runs on a Sun SparcStation. Speech is input using a good microphone and the built-in 8Hz A/D converter. A simple template-matching word-spotter produces a lattice of word hypothe-

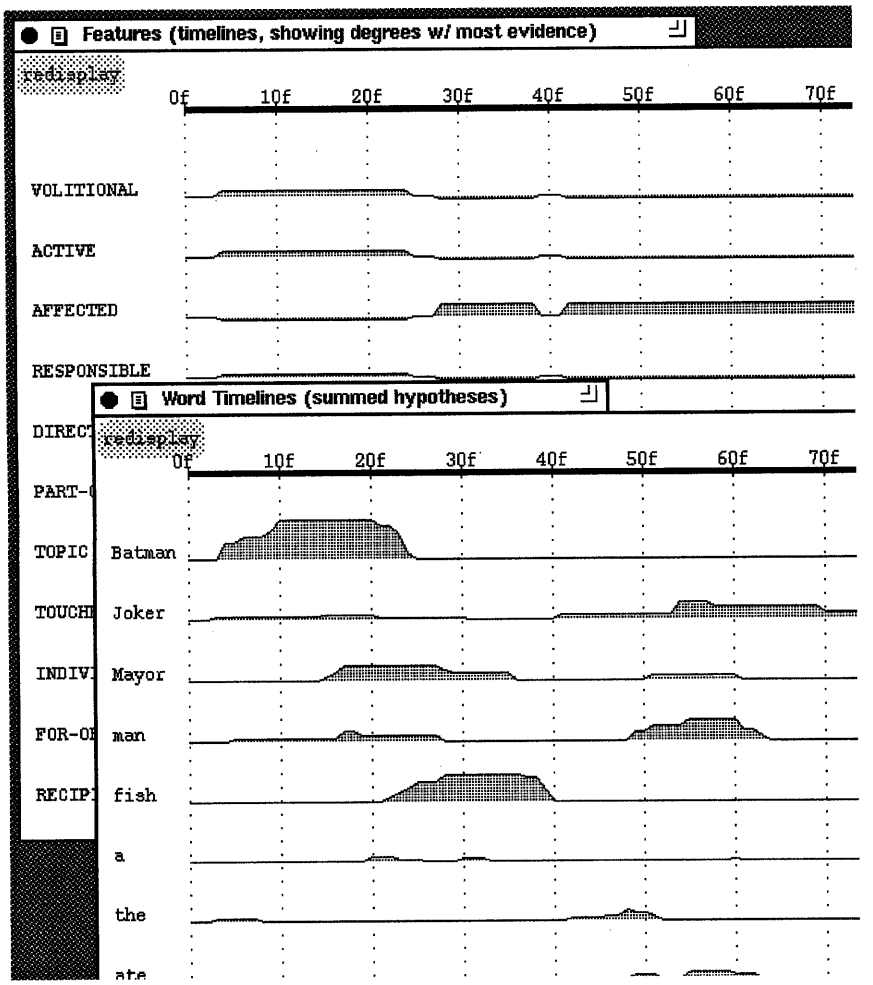

Figure 4: The participatory profiles hypothesized at each timepoint. This was computed from the construction hypotheses (Figure 3). For example, the high value for *affected* towards the right is due to two factors. The first is the that direct objects are typically affected, combined with the presence of a some fairly highly ranked *transitivec* hypotheses whose second constituents (the direct object) span this region. The second is the lack of significant countervailing evidence from Subject-Predicate hypotheses.

Figure 5: The distribution of evidence over time, for each word. This was computed from the word hypotheses (Figure 2) by summing, for each timepoint, the scores of all hypotheses for that word.

ses, each consisting of word name, start point, end point, and score. This spotter finds words in continuous speech, slowly, with low accuracy. The parser operates as discussed above.

From the clues output by the parser, a simple semantic interpreter applies knowledge about good and evil to come up with an interpretation suitable for an emotional response. For example, it knows that those *responsible* for *rescues* are *good-guys*, so if, for a given input, the parser gives high scores to

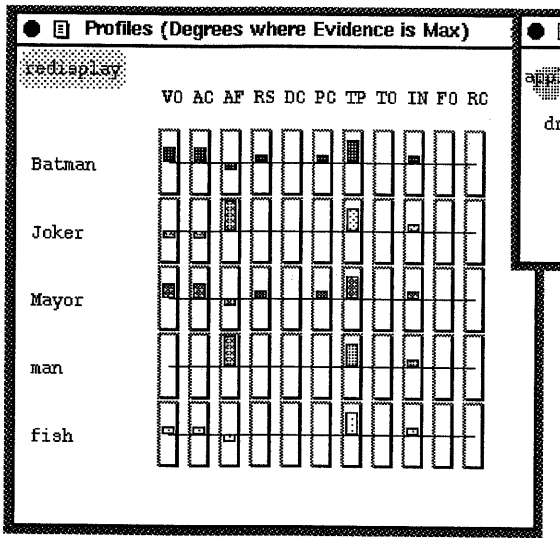

Figure 6: Summary of the computed profiles for words. These are computed by, for each word and each feature, integrating over time the product of the feature score (Figure 4) and the word score (Figure 5). For example, the high value for *affected* (the 'AF' column) for *"man"* is due to the fact that the bulk of the evidence for man is in the timespan where *affected* is highly scored.

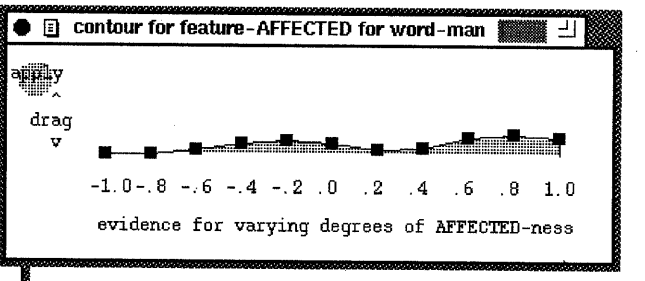

Figure 7: the detailed breakdown of evidence for *man* being *affected*, to various degrees. The peak value falls at the degree .8; which is why .8 is the value for AF shown in Figure 6.

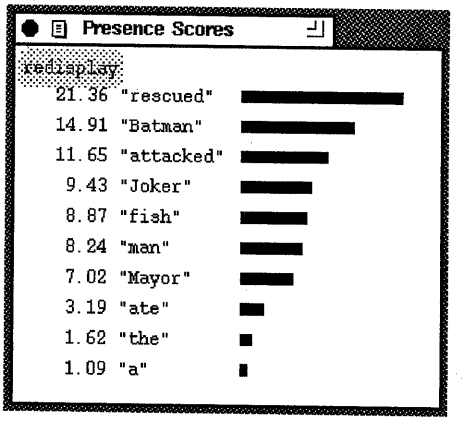

Figure 8: The computed "presence scores" for each word. These were obtained by summing, for each word, the scores of all its hypotheses (Figure 2).

the clues *the-mayor-is-responsible* and *the-rescue-concept-is-present*, then the interpreter will give a high score to the conceptual hypothesis *the-mayor-is-acting-as-a-good-guy*. The semantic interpreter also has some basic knowledge about personalities; for example, it knows that *batman* is generally a *good-guy*. It can then use such facts to rescore clues, and thus provide feedback to the parser and eventually the word spotter.

Since the semantics of this domain are rather trivial, there is also the facilty for human intervention, to provide additional semantic feedback — a window interface enables the developer to adjust the scores of the various conceptual hypotheses and observe how the effects propagate through the system. (This will be the focus of the video).

5 Example

Figures 2–9 illustrate how the system works. The input was *"Batman rescued the mayor"*, spoken fairly sloppily.

Figures 6 and 8 constitute the set of clues (linkages are not yet implemented). From these, the semantic interpreter arrives at the interpretation shown in Figure 9. If these scores are adjusted, as a result of semantic interpretation or intervention by the experimenter, the scores for upstream hypotheses are recomputed; this feedback is basically just the inverse of the process illustrated above.

For example, based on the knowledge that the Joker is generally not a good-guy, since this input involves a *rescue*, the score for the hypothesis *joker-is-responsible* will be decreased. Then, since there is evidence for *responsibility* for words appearing in the region from frame13 to frame24 (see Figure 4), there is evidence against any hypotheses which place

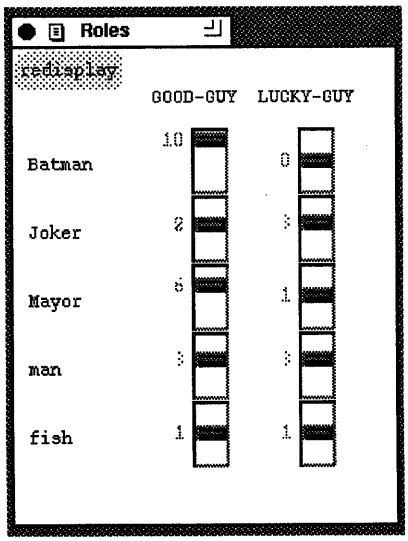

Figure 9: The semantic roles inferred from the participatory profile (Figure 6) and word presence (Figure 8) hypotheses. For example, the high score for *Batman* on the *good-guy* scale is largely due to the application of the knowledge that those *responsible* for *rescues* are *good-guys*.

the word *"Joker"* in that time region.

As another example, based on the knowledge that Batman is likely to be a good-guy, since this input involves a rescue, it is likely that he is *volitional*. Given that the evidence for the word Batman appearing is clustered around frames 13 to 24, the system concludes that there is top-down evidence for placements of the Subject-Predicate construction where the first constituent falls in that region (for example, the top hypothesis in Figure 3), because English subjects are typically volitional.

6 Open Questions

This system has demonstrated that use of a lightweight parser allows the application of semantic feedback for speech understanding. Many questions remain.

For one, it has not yet been show whether tight coupling is advantageous. Using this system, I plan to examine this, by measuring the advantage, if any, of providing semantic feedback to the word spotter, rather than than simply using it to rescore the n-best sentence hypotheses.

The details of the algorithms for lightweight parsing, in particular, for the computation of the scores for various hypothesis, need more work.

It is also not clear whether this particular approach to lightweight parsing will scale up. Specific concerns include: For larger vocabularies, will it still be okay to just sum up scores with a timeline (rather than assembling coalitions of compatible hypotheses)? For larger lattices and more constructions, will the synergy among construction hypotheses turn into cacophony? How can the various weights and probabilities involved with construction hypotheses be learned?

References

Bates, Madeline, Robert Bobrow, *et al.* (1993). The BBN/Harc Spoken Language Understanding System. In *1993 IEEE ICASSP*, pp. II–111–114.

Fillmore, Charles J. (1988). The Mechanisms of "Construction Grammar". In *Berkeley Linguistics Society, Proceedings of the Fourteenth Annual Meeting*, pp. 35–55.

Jurafsky, Daniel, Chuck Wooters, *et al.* (1994). Integrating Experimental Models of Syntax, Phonology, and Accent/Dialect in a a Speech Recognizer. In *AAAI Workshop on the Integration of Natural Language and Speech Processing.*

Martin, Charles E. (1989). Case-based Parsing. In Christopher K. Riesbeck & Roger C. Schank, editors, *Inside Case-based Reasoning*, pp. 319–352. Lawrence Erlbaum Associates.

Seneff, Stephanie (1992). TINA: A Natural Language System for Spoken Language Applications. *Computational Linguistics*, 18(1):61–86.

Ward, Nigel (1992). An Alternative to Deep Case for Representing Relational Information. In *Proceedings 14th COLING.*

Ward, Nigel (1993). On the Role of Syntax in Speech Understanding. In *Proceedings of the International Workshop on Speech Processing*, pp. 7–12. also Gijutsu Hokoku SP93-76, IEICE, Tokyo, 1993.

Ward, Nigel (1994). *A Connectionist Language Generator.* Ablex.