INTEGRATING SPEECH AND NATURAL-LANGUAGE PROCESSING

Robert Moore, Fernando Pereira, and Hy Murveit
SRI International
Menlo Park, California 94025

ABSTRACT

SRI has developed a new architecture for integrating speech and natural-language processing that applies linguistic constraints during recognition by incrementally expanding the state-transition network embodied in a unification grammar. We compare this dynamic-grammar-network (DGN) approach to its principal alternative, word-lattice parsing, presenting preliminary experimental results that suggest the DGN approach requires much less computation time than word-lattice parsing, while maintaining a very tractable recognition search space.

INTRODUCTION

We report on an innovative and highly effective architecture for integrating speech and natural-language (NL) processing. This architecture takes full advantage of the linguistic constraints supplied by an NL processor, yet it maintains a very tractable recognition search space and appears to require dramatically less computation by the NL processor than do other approaches.

It is useful to begin by recalling why one would want to integrate speech and NL processing. There are two primary reasons why this is desirable. First, there are many applications of spoken-language processing that require not just recognition but understanding, such as spoken-language systems for database query and other computer interface applications. Second, constraints on natural language can be used to reduce the speech recognition search space and therefore improve recognition accuracy.

To take an example from our own system, one of the sentences in the 1987 DARPA resource management speaker-independent test set is the following:

> What is the ETA at her destination of Fanning?

Without any source of grammatical constraints (perplexity = 1000), one version of the SRI system recognizes this sentence as:

> Why added ETA at her destination of Fanning.

This hypothesis contains three errors out of nine words in the original sentence. With the fairly modest degree of grammatical constraint represented by a perplexity-510 natural-language grammar, however, our system was able to recognize this sentence with no errors.

The point is that even though the incorrect hypothesis represented a better phonetic and phonological match to the input signal according to the models used by our system, with an NL grammar the system was able to rule out the incorrect hypothesis and choose the correct, although lower-scoring, hypothesis. While this is, of course, a carefully selected example of our system's performance, it is a good illustration of what we are trying to acheive.

OTHER APPROACHES TO SPEECH/NL INTEGRATION

Prior to the work reported here, there have been two predominant approaches to integrating speech and NL processing. The first is the serial connection of speech and NL processors, and the second is word-lattice parsing.

In the serial approach, the speech recognizer makes its best guess as to what word string was uttered, and the NL processor parses and interprets the word string as if it were text. This approach has the advantages that it is simple to implement, and it is relatively fast because neither process increases the search space of the other. Unfortunately, it takes no advantage of the NL constraints embodied in the NL processor to guide the recognizer, so we do not acheive the kind of integration we are looking for.

In word-lattice parsing, this problem is addressed by having the recognizer refrain from making any final decisions as to what string of words has been uttered. Instead the recognizer passes to the NL processor a word lattice—a set of word hypotheses and their acoustic match scores—and the NL processor has to find the best scoring path, through the word lattice, that constitutes a well-formed utterance according to the constraints the NL processor knows about.

Word-lattice parsing thus succeeds in applying NL constraints to aid recognition, but it results in extremely long processing times, at least in current implementations. We have identified a number of factors that we believe are responsible for this. First, the parser must deal with a word lattice containing thousands of word hypotheses rather than a string of just a few words. This, in effect, amounts to a massive degree of lexical ambiguity, giving the parser many more possible analyses to consider than in text parsing. More particularly, the parser must deal with a large degree of word-boundary uncertainty. Normally, a word lattice of adequate size for accurate recognition will contain dozens of instances of the same word with slightly different start and end points. A word-lattice parser must treat these, at least to some extent, as distinct hypotheses. The best approach to this problem proposed so far (Chow and Roukos, 1989) seems to be to associate with each word or phrase a set of triples of start points, end points, and scores. Each possible parsing step is then performed only once, but a dynamic programming procedure must also be performed to compute the best score for the resulting phrase for each possible combination of start and end points for the phrase.

In addition to the parser having to do extra work, word-lattice parsing requires extra work of the recognizer. Because word hypotheses are put together to form paths by the parser rather than the recognizer, the recognizer must treat each word starting at each point in the input as an independent hypothesis. Thus, much of the efficiency of the dynamic programming methods used in recognition systems is lost, and the recognizer faces a much larger search space than if it were running with no grammar or a simple finite-state grammar. Moreover, if the recognizer is using a pruned search to generate the word lattice, as it must to have any hope of practicality, it cannot take advantage of grammatical constraints on acceptable paths in making its pruning decisions. The recognizer is unable to make use of the fact that a word hypothesis aligns with only low-scoring acceptable paths to help prune that hypothesis.

OUR APPROACH: DYNAMIC GRAMMAR NETWORKS

In view of the efficiency problems of word-lattice parsing, we have developed an alternative approach based on the concept of dynamic grammar networks. In this approach, we use an NL parser to incrementally generate the grammar-state-transition table used in the standard hidden-Markov-model (HMM) speech-recognition architecture. In an HMM speech recognizer, a finite-state grammar is used to predict what words can start in a particular recognition state and to specify what recognition state the system should go into when a particular word is recognized in a given predecessor state. The Viterbi algorithm is used to efficiently assign a score to each recognition state the system may be in at a particular point in the signal.

In HMM systems, the finite-state grammar is represented as a set of state-word-state (or state-word-state-word) transitions. Any type of linguistic constraints can, in fact, be represented as such a set, but for a nontrivial NL grammar the set will be infinite. Even a useful finite approximation would be too large to

compute given current hardware, and too large to store in current memories that would be fast enough for practical applications of speech recognition.

Our approach to this problem is to compute the state-transition table needed by the HHM system incrementally, generating just the portion necessary to guide the pruned search carried out by the recognizer for a particular utterance. When the recognizer is started up, it contains an initial state and the words that are possible in that state according to the NL grammar. When a word is successfully recognized without being pruned, the recognizer sends the word and the state it started in to the NL processor, which returns the states that can follow and a specification of what words can start in each state.

In our NL processor, we use a stack-based unification-grammar parser incorporating left-context constraints. The parser is roughly equivalent to an LR(0) parser that generates LR(0) item sets on the fly. The recognition states are derived from parser configurations, so that when the recognizer sends a state and a word to the parser, the parser can decode the state and advance the resulting parser configurations by the recognized word. The parser then encodes the resulting configurations and returns a specification of a set of state-word pairs to the recognizer.

Although in this architecture the recognizer rather than the parser has control of the search, the parse or parses of the recognized string can easily be recovered for further NL processing. The backtrace information the recognizer uses to recover the recognized string also contains the recognition states along the backtrace path. Because these states encode parser configurations, it is possible to recover the parses of the recognized string with no additional search.

It should be emphasized that this approach is computationally completely equivalent to word-lattice parsing, although it seems to be much faster. One can think of the paths through a word lattice as representing the recognition search space. In word-lattice parsing the entire search space is generated, encoded in the word lattice, and NL constraints are brought to bear afterward to prune out the paths that violate those constraints. In the dynamic-grammar-network approach, the same constraints are brought to bear as the search space is being generated, so the paths that violate the constraints are never produced. In either case, the final set of paths allowed is the same.

This architecture addresses the major efficiency problems we pointed out in word-lattice parsing. Because the recognizer does all the tracking of input positions and scores, the parser has no need to deal with word-boundary uncertainty. In fact, no information about word boundaries is even passed to the parser. This is possible because the parser is stack rather than position based. Because NL constraints are applied at every state, the recognition search is confined to the states that arise in the grammar rather than treating every point in the input as a distinct recognition state. Thus, the Viterbi search can collapse any word hypotheses that begin in the same state and end at the same point, without having to keep track of their starting points. Finally, because the recognizer is following paths that are guaranteed to meet the NL constraints, its pruning can take advantage of that information to help prune out word hypotheses that continue only low-scoring paths.

In addition to addressing the efficiency problems of word-lattice parsing, the dynamic-grammar-network approach has a number of other important advantages. First, it presents a very clean interface between the recognizer and the NL processor, preserving the structure of existing recognition systems. Thus, everything that has been learned about making HMM systems efficient can be applied in this design. This carries over to implementations of the HMM architecture in special-purpose hardware currently under development. The only modification of such designs that will be necessary is to transmit to the parser what recognition states have been reached, and to allow the parser to download state transitions into the HHM system's grammar memory.

A particular advantage of our implementation of the dynamic-grammar-network approach is that it is based on unification grammar, which is not only a superior formalism for encoding syntactic constraints on natural language, but which, as our prior work has shown, can be used to encode semantic constraints as well. We believe this is of critical importance, because syntax alone seems to be fairly limited in its ability to constrain recognition, but semantic constraints seem to be far more restrictive. These can also be expressed in our unification framework and applied by the unification parser in our current implementation.

Finally, because recognition and NL processing are combined so tightly, it appears that our architecture

Pruning Threshold	Med/Mean Sentence Length	Med/Mean Parsing Time (sec)	Med/Mean Active Hyp per Frame	Cumulative Word Accuracy
400K	6.5/7.1	63/102	539/592	85.5%
500K	6.5/7.1	129/310	852/1523	87.8%

Table 1: Experimental Results

is ideally suited to exploring the interactions of prosody and phonology with syntax. This should provide yet another source of constraint on the recognition problem, further improving recognition accuracy, and it should also provide prosodic and phonological information to help resolve syntactic ambiguities. It appears that many of these constraints can be encoded in the unification formalism in our system.

RESULTS OF INITIAL EXPERIMENTS

In the preceding sections, many theoretical arguments have been advanced as to why the dynamic-grammar-network approach ought to be superior to its possible competitors. Without empirical support, however, those predictions would be of little significance. We have built an initial implementation of our architecture that, in fact, seems to bear out those predictions in a fairly dramatic fashion. The system has been tested on twenty-four sentences chosen from the 1987 DARPA resource management speaker-independent test set, using an 885-word subset of the standard 1000-word vocabulary. The perplexity of this test set was measured to be 510 with respect to the vocabulary and NL grammar used in the test.[1] Currently our grammar covers about half the sentences in the resource management database. We do not expect the efficiency of our system to be seriously affected as coverage increases, however, because we already cover the constructions that lead to most of the complexity in natural-language parsing: conjunction, relative clauses, WH-questions, prepositional phrases, and other postnominal and postverbal modifiers. On the test set used in this experiment, SRI's recognizer scores 82.6 percent word accuracy with no grammar (perplexity-1000), and 97.2 percent with a perplexity-60 word-pair grammar.

The results of our experiments are summarized in Table 1. The table presents summaries of the results of running all twenty-four test sentences with two different pruning thresholds. The smaller number represents tighter pruning. First, we note that at both levels of pruning there appears to be a significant improvement in recognition accuracy over the recognizer with no grammar. There was only one sentence of the twenty-four on which the no-grammar recognizer performed better than the run with the 400K pruning threshold, and no such sentences for the run with the 500K threshold. The run with the 500K threshold produced a 30 precent reduction in the word-error rate. We believe that the really important results of this test, however, are the parsing times and the recognition search space size, the latter being measured by average number of hypotheses active in a given frame. We show both median and mean values for these measures.

The parsing times are for a parser running in Prolog on an 8-megabyte Sun 3/60 workstation. At the 400K pruning threshold, the median parsing time per sentence was just over one minute, and at 500K, just over two minutes. This compares with two hours or more for word-lattice parsing experiments reported elsewhere. Thus, these times represent an improvement of approximately two orders of magnitude over previous results with word-lattice parsing. The recognition search space size is an equally significant measure of the success of this architecture. The median number of active hypotheses per frame was under 1000 in both runs. Because the real-time recognition hardware SRI is currently designing and building will ultimately have a capacity to process 6000 active hypotheses per frame, our architecture should easily support real-time recognition with multithousand word vocabularies.

[1] Although the test set is small, it does represent (with the exception of one sentence discussed below) all the sentences we have had an opportunity to test our system on to date. Thus it is not specially selected to show our system to advantage.

The parsing times we have acheived, as good as they are compared to previous results, still require more than an order of magnitude improvement for real-time performance. In addition, the fact that the mean times were so much higher than the median is an indication that on difficult sentences, the parsing time can become extraordinarily long. These turn out to be sentences where the pruning thresholds fail to reduce the hypotheses considered to a reasonable number. The parser then simply bogs down in the resulting mass of data.[2] We believe this problem can be addressed by more sophisticated pruning methods.

Getting the median parsing time down to real-time levels can be addressed in a number of ways. There are many algorithmic improvements we are already planning to try. Moreover, simply moving to faster hardware will decrease parsing time substantially, and another significant speedup would be obtained by recoding in a lower-level language such as C. We believe that with these improvements and the advent of real-time recognition hardware, it is not unreasonable to hope for a complete real-time system within the relatively near future.

SUMMARY

We believe the results reported in this paper have demonstated:

- A dramatic improvement in parsing times for integrated speech/NL systems
- A highly tractable recognition search space
- Significant improvement in recognition performance with natural-language grammars
- An architecture that can exploit existing recognition system designs and prospective real-time hardware with only minimal modifications
- A clear path toward including important semantic constraints on recognition, and possibly prosodic-phonological/syntactic constraints as well

These results suggest that the dynamic-grammar-network approach to the integration of speech and NL processing may well be the key technology needed to produce true spoken-language systems with real-time performance.

Acknowledgments

In addition to the authors, significant contributions to the work reported here have been made by Mary Dalrymple, Mike Cohen, and Mitch Weintraub. This work has been supported by SRI International internal research and development funds.

References

Chow, Y. L., and S. Roukos (1989) "Speech Understanding Using a Unification Grammar," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, Scotland (May 23–26, 1989).

[2]Indeed, a twenty-fith sentence was originally included in our test set, but had to be discarded because the 8-megabyte workstation the parser was running on was too small to get the entire working set for this sentence in memory at the same time. Had that sentence been included, the mean parsing times would undoubtedly have increased significantly, but the median times would have changed only slightly.