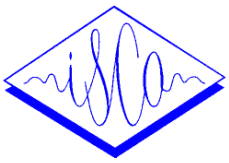

2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

A CSR-NL Interface Architecture*

Douglas B. Paul

Lincoln Laboratory, MIT
Lexington, Ma. 02173

Abstract

Spoken Language Systems will require integration of continuous speech recognition and natural language processing. This is a proposed architecture for an interface between a continuous speech recognizer (CSR) and a natural language processor (NLP) to form a spoken language system. Both components are integrated with a stack controller and contribute to the search control. The architecture also allows a "Top-N" mode in which a "first part" outputs a list of top N scored sentences for post-processing by a "second part". An additional use for this architecture might be NLP evaluation testing: a common simulated CSR could be interfaced to each site's NLP to provide identical testing environments.

1. Introduction

This is a description of an architecture for an interface between an acoustic matcher, such as a Hidden Markov Model (HMM) CSR, and a language model component such as a stochastic language model or an NLP. Its purpose is to allow independently developed CSR and NLP systems to be interconnected by a well specified and well structured interface.

This architecture provides for two modes of operation: integrated and decoupled. In the integrated mode, both the CSR and the NLP contribute to the search control. If (or when) the CSR and NLP technologies are sufficiently mature, this will probably be the preferred mode. The decoupled mode allows the CSR component to output a list of possible sentences with acoustic match likelihoods. The NLP can then process this list as it sees fit. Since information flow in the decoupled mode is strictly feed-forward, no NL information is available to help constrain the search in the CSR component.

The architecture contains overall control structure and interface definitions. The resulting system consists of a combined stack-controller/CSR (SC-CSR) and NLP interconnected by UNIX pipes. The basic algorithmic constraints required by this interface are fairly mild: the interprocess interface uses UNIX pipes, and both the CSR and NLP components operate left-to-right on their respective input data. (A more detailed but obsolete description of this interface can be found in [4].)

*This work was sponsored by the Defense Advanced Research Projects Agency. The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government.

2. The basic system concept

This architecture was derived by partitioning the stack decoder [1, 5] into three parts:

1. A stack decoder controller.
2. A CSR capable of evaluating the probability of the acoustic data for a given left sentence fragment.
3. An NLP capable of evaluating the probability of a given left sentence fragment.

The basic system operation is:

1. Initialize the stack with a null theory.
2. Pop the best (highest scoring) theory off the stack.
3. if(end-of-sentence) output the sentence and terminate.
4. Perform acoustic and language-model fast matches [2, 3] to obtain a short list of candidate word extensions of the theory.
5. For each word on the candidate list:
   (a) Perform acoustic and language-model detailed matches to compute the output log-likelihood for the new theory.
      i. if(not end-of-sentence) insert into the stack.
      ii. if(end-of-sentence) insert into the stack with end-of-sentence flag = TRUE.
6. Go to 2.

Top-N (N-best) mode is achieved by delaying termination until N sentences have been output.

3. Detailed Concepts

3.1 The likelihood function

Log-likelihoods are used to evaluate theories. The stack likelihood function should also include some extra control parameters:

$$theory\_likelihood = CSR\_likelihood + \beta * NLP\_likelihood + \gamma * nr\_words \quad (1)$$

where β is a grammar weight, γ is a word insertion penalty, and nr_words is the number of words in the theory. Beta controls the relative weights of the acoustic and grammatical evidence. Gamma controls the relative number of insertion and deletion errors. This *theory_likelihood* is used by the stack to control the search [5].

3.2 Partial theory memory

Memory-less CSR and NLP components are inefficient because they require recomputation of the imbedded left sentence likelihoods. Thus, both the CSR and the NLP will cache the partial theories and the information required to efficiently compute any extensions of those theories. Theory identification numbers (handles) will be used to identify the theories. The theory identifiers will be 1-1 with the theories.

3.3 Stochastic grammars

Probabilities are the common language for communication between the two modules and the search control. Thus, grammars which give the probability of a full or partial sentence provide much more information to the combined CSR-NLP system than grammars which just accept or reject a sentence. The control scheme used in this architecture is tolerant: it can handle full probabilities, branching factor based probabilities, or just acceptance-rejection "probabilities" (ie. 1's or 0's). Presumably, the the more accurate the probabilities, the better the overall performance.

One ad-hoc method for adding probabilities to an accept-reject language model is to use probabilities from a simple stochastic language model (e.g. N-gram) on the word sequences allowed by the accept-reject language model.

3.4 Top-N mode

The stack controller can continue to output sentences in decreasing likelihood order. This might be used in decoupled mode to allow later reevaluation of the top-N theories by an expensive but more accurate language model or NLP.

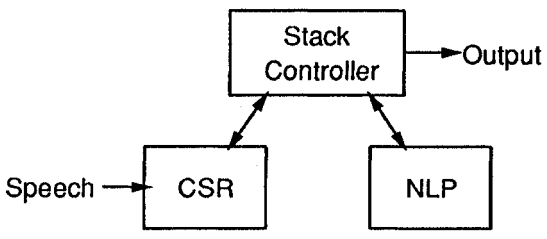

Figure 1: Conceptual Architecture

3.5 Speech Understanding

The "normal" output of this system is the best word sequence which matches the acoustic and NL constraints. In addition, a mechanism is included for the NLP to output the meaning of the recognized sentence. This meaning will be expressed as text (ie. ascii characters to make it machine independent), but its format is undefined by this specification. This will allow the NLP to feed an interpretation or a parse tree to a later module for execution. For example, a database query system might output in a database query language or it might output a parse tree for later interpretation into a database query.

3.6 Control

The stack is the sole controller of the system. It sends out a request to a slave and waits for a reply. Either slave (ie. the CSR or NLP) may in turn make a request of a helper, but any such helpers must be slaves of the CSR or NLP. Neither the CSR or NLP nor any helpers may initiate any action involving the stack.

3.7 Integration of the Stack and the CSR

If the system were configured into three separate modules as shown in the figure, it would require excessive communications overhead. The communications with the NLP are simpler than with the CSR since time registration is not an issue for the NLP. Because the CSR must actually return likelihood distributions to the stack, the stack and the CSR are integrated into a single stack-controller CSR module (the SC-CSR) to remove the higher bandwidth channel. This causes no change in the control structure: the stack is still the sole master and the CSR and the NLP are still its slaves. This also causes no change in the NLP interface.

4. The Architecture

Logically, the architecture consists of the three parts listed above: the stack controller (SC), a CSR, and an NLP. (As mentioned above, the stack controller will be combined in a single process with the CSR, but will have the same functionality as if the two were separate.) The SC-CSR process will communicate with the NLP process via UNIX pipes. Therefore, the two processes need not be written in the same language and need not even be running on the same machine. The NLP will receive its commands on the I/O channel "stdin" and reply on I/O channel "stdout" (Stderr will retain its usual function.)

To minimize the communications overhead, the request for detailed matches may be batched in groups which are extensions of the same theory. Thus the block of commands will be sent from the SC to to the NLP and the replies will be expected as a block (in corresponding order) when the NLP is finished. Similarly, a list of theories can be purged in a single command. This will be particularly important when two separate machines are used.

5. The data format

The messages will consist entirely of text in order to make them machine and language independent and easy to debug. Both processes will cache partial theories which will be identified by a positive unordered integer handle (label). Handle "0" will be the null theory. Communications between the stack-CSR and the NLP will be in a command-reply format. A partial list of exchanges is shown in Table 1.

6. Discussion

This interface has been implemented in the Lincoln stack decoder[6] using both word-pair and N-gram language models. The word-pair language model is an accept-reject model and the N-gram language models are stochastic language models. Both of these models have very efficient detailed matches and therefore no language model fast match was implemented. The interface has been tested on a single machine and also between two machines connected by a local-area network and works as expected.

7. Acknowledgement

An advisory committee of both NL and CSR people aided the author by reviewing the proposed architecture from both viewpoints. The committee members were: Janet Baker (Dragon Systems), Charles Hemphill (TI), and Lynette Hirschman (MIT). The comments of these committee members were very helpful to the author and their membership does not imply agreement with the provisions of this specification.

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition" PAMI-5, no 2, March 1983.

[2] L. R. Bahl, P. S. Gopalakrishnam, D. Kanevsky, D. Nahamoo, "Matrix Fast Match: A Fast Method for Identifying a Short List of Candidate Words for Decoding," ICASSP 89, Glasgow, May 1989.

[3] L. S. Gillick and R. Roth, "A Rapid Match Algorithm for Continuous Speech Recognition," Proceedings June 1990 Speech and Natural Language Workshop, Morgan Kaufmann Publishers, June, 1990.

[4] D. B. Paul, "A CSR-NL Interface Specification," Proceedings October, 1989 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, October, 1989.

[5] D. B. Paul, "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition With a Stochastic Language Model," ICASSP 91, San Francisco, March 1992.

[6] D. B. Paul, "The Lincoln Large-Vocabulary HMM CSR," Proceedings October, 1992 DARPA Speech and Natural Language Workshop, Morgan Kaufmann Publishers, February, 1992.

Stack-CSR command	NL reply	Explanation
ready ver_nr	ok	Ready, version nr
reset	ok	Reset NLP
<old-id> <new-id> <word> <blank-line>	 log-prob <blank-line>	single detailed match: append word to old-id assign to new-id return log-prob
<old-id> <new-id1> <word1> <new-id2> <word2> <blank-line>	 log-prob1 log-prob2 <blank-line>	batched detailed match
purge <id> <blank-line>	 ok	purge single theory
purge <id1> <id2> <blank-line>	 ok	purge theory list

Table 1. A partial list of stack-CSR commands and NL replies.