Session 4: System Implementation Strategies

Roberto Bisiani, Chair
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This is both a summary of what happened at the session and an introduction to the papers. Although the opinions expressed here are as balanced as possible, they might reflect my bias, for which I apologize in advance.

The session was targeted towards the issues that are raised by the implementation of complete, real-time systems. It seems that the Speech Community has finally realized the importance of demonstrating fully usable real-time systems. Two definitions of real-time seemed to be acceptable to the workshop participants. A system is *real time* either if:

- it can keep-up with continuous input or if
- it returns a fully parsed sentence within 200ms from the end of the utterance.

The issues in implementing usable systems are:

- recognition speed;
- development cost and time;
- production cost.

The papers in the session concentrated on the first two issues. It was very nice to see that two papers out of three provided non-trivial speed improvements mainly with software techniques.

The BBN paper, the first to be presented, hinged on a number of algorithmic improvements. The most notables being the reduction of the number of transitions in a grammar by means of *zero states* and the use of a suboptimal but fast Forward-Backward Search. It is difficult to precisely evaluate the relative improvement figures shown in the paper because they are measured with respect to the speed of an initial system of unknown (programming) quality, Nevertheless, the algorithmic speed improvements were substantially larger than the improvements due to the hardware (a state-of-the-art i860 board).

The second paper, presented by Dragon, hinged on a technique, called *rapid match*, that cuts the number of hypothesis during the search by limiting it to a subset of the possible words. This technique makes it possible to implement useful and impressive recognition systems on garden-variety 386-based personal computers. As with the previous paper, the advantages provided by better hardware were much less impressive than the advantages made possible by clever algorithms.

A completely different approach was presented by SRI and Berkeley. The paper describes a custom machine that implements a heavily pipelined Viterbi search. Custom chips and a large amount of memory make up the bulk of the machine. The performance, at least on paper, is about two orders of magnitude better than the performance of current general purpose systems. Although this gap might be reduced to one order of magnitude by the introduction of new general purpose systems, the performance of this system is potentially very impressive. The audience had questions on technology (MOSIS CMOS) and on availability (sometime towards the end of the year).

At the end of the session the chairman gave a brief progress report on the PLUS system being developed by CMU. This system was not described at the workshop because it has already been described in computer architecture papers. PLUS is a distributed-memory multiprocessor composed of mesh-connected nodes. Each node contains a Motorola 88000, static and dynamic memory, and circuitry to make the local memory visible to all the other processors as if it were local to them. Systems with 1 to 64 nodes are possible. Each system is connected to a supporting workstation through its SCSI bus, facilities for input/output of digitized speech are provided. Construction of PLUS is proceeding and a few nodes will be running by the end of the summer.

This, I believe, was a very positive session. It showed us that it will soon be possible to implement **in real time**:

1. small but non-trivial tasks on commercial hardware;
2. complex tasks that require fast search on custom hardware;
3. full complex tasks, including natural language processing and data base search on semi-custom hardware.

All these solutions will cost no less than a medium-size workstation. Should we start worrying about how to use algorithmic improvements and technology to build much **cheaper** systems?