MAJOR FINDINGS

Since the main component of this project is the development and dissemination of speech recognition technology, we did not expect to generate a significant list of technology-related research accomplishments in the first year of the project. Nevertheless, we have begun some interesting research as peripheral activities. These research topics include the use of Support Vector Machines for improved acoustic modeling, the study of the influence of context-sensitive word duration models on conversational speech recognition performance (a step in the direction of introducing prosodics into the speech recognition problem), and implementation of a new segmental Baum-Welch training algorithm (preliminary results for these approaches look promising; detailed results should be available by December 1999). The fact that such research can be easily performed with our system supports our contention that the system is extensible.

With respect to the core technology component of the program, we believe we have delivered a decoder that is extremely efficient for conversational speech recognition, and is competitive with state-of-the-art. Decoding time and memory requirements are within the reach of standard PC-class computers. This is important in the context of this program because it will increase access to this technology by allowing smaller research labs to be able to use the system with fairly modest computing environments. To move to larger domains than conversational speech, such as broadcast news, we have developed a dynamic language modeling capability that caches large language models to decrease physical memory requirements. We have also demonstrated that porting of the system to any gcc compliant platform is fairly easy. The only outstanding issue is wide character support (Unicode) under Linux. Once Linux compilers catch up (expected in Fall’99), our cross-platform support problems should be minimal.

Foundation class development has proceeded using a model similar to Java, but adapted to the demands of speech research. We have found it extremely useful to abstract the user from the details of the operating system through the use of our system classes. These handle all low-level interactions with the operating system, and centralize many tasks such as memory management, file management and I/O. The next level above the system classes, the math classes, provide the user with basic data type building blocks. Here we have followed an STL model, and have demonstrated that a mixture of templates and fixed classes are an optimal way to compromise between the needs of low-level programmers to see physical data types (such as short integers) and the needs of high-level programmers to be able to build generic math objects (for example, a matrix of signals). Templates have only become practical with recent releases of C++ compilers.

Web-based dissemination of project information has proven to be a mixed bag. Unfortunately, a significant percentage of people interested in our technology and resources appear to still have limited Internet bandwidth and access. Hence, the demand for small distributions that can be downloaded via slow modems still exists. This severely limits what we are able to accomplish in the way of on-line documentation, interactive applets, and distribution of toolkits including enough data to run a reasonable experiment. Our anonymous CVS server has been very useful in that it allows users only to download pieces of the code that have changed — thereby reducing the amount of data one needs to download to remain current.

The remote job submission facility, though extremely unique and impressive, is not receiving the initial traffic we had expected. Users still seem to prefer to download the package and build the demos on their local machines. We hope to improve the visibility of this facility by enhancing and streamlining the user interface in the next year of this project.