08/15/00 — 08/14/01: MAJOR FINDINGS

In the third year of this project, we continued conducting our annual workshops. In January 2001, we hosted 14 visitors for a software design review. In May 2001, we hosted 24 participants for our one-week training workshop. Several collaborations resulted from our previous workshops, and it appears the same will be true this year as well. A promising trend this year has been a significantly increased level of interest by commercial users in the software and technology.

A new problem-tracking tool was introduced that has greatly impacted our software design process. This tool, called Varmint, is publicly available and is part of our public domain software distribution. Varmint allows us to track the life-cycle of a bug, and is particularly useful in a multi-programmer environment in which several people may touch a bug during its life-cycle. Varmint is styled after several commercial tools, and was designed based on several common models of bug tracking used by information technology professionals. The most significant benefit of using this tool is that programmers are explicitly conscious of the bugs for which they are responsible, and have a clear understanding of the priority of these bugs with respect to the current software release schedule. This creates the proper atmosphere of accountability required to make sure releases are clean before they are made.

We also released a version of our job submission applet that uses Java servlets. While the servlet programming environment is still rapidly evolving, servlets have been instrumental in allowing our interfaces to be sufficiently powerful. For example, users can now browse our filesystems or their own filesystems using the same interface, and can collect a diverse set of files for processing. These features allow users to benchmark their local implementations against our reference implementations. One windfall from this has been a reduction in support requests on trivial issues such as file formats because users can now debug this themselves using the applet.

We have continued to release a large amount of supporting materials on our web site that are relevant to our mission of providing comprehensive conversational speech recognition tools. We now deliver high-quality transcriptions for 500 hours of Switchboard conversational speech data, and offer acoustic models trained on this data. ISIP resources are frequently referenced at major speech recognition forums such as the Department of Defense’s Speech Transcription Workshop (also known as the Hub 5E workshop). This year, we added phonetically transcribed data originally developed at the International Computer Science Institute at the University of California, Berkeley. This data, along with our word-level transcriptions, are provided in a format that makes it easy to use for investigations into phonetic-level performance of speech recognition systems. This data is being used by several students in their dissertation research.

Our on-line support and tutorial materials continue to grow as we collect more feedback from our users. Our support line averages 5 support requests per day, about the same level of activity we experienced in the previous year of this project. Typically one of these requests will be nontrivial and require a more measured response. We estimate that we spend approximately 20 hours per week performing direct support of the software through the on-line help service. The user base for our software appears to have stabilized at about 175 users.

We have also in the past year begun to graduate our first thesis-option students who contributed to this project. Thesis topics have ranged from advance pattern recognition techniques (Support Vector Machines) to experimental topics such as the influence of transcription errors on performance. These theses are the first work to make extensive use of the ISIP tools.

08/15/99 — 08/14/00: MAJOR FINDINGS

In the second year of this project, we introduced two pivotal activities in the project: annual workshops and a rigorous software distribution process. The workshop activities are proceeding smoothly. In January 2000, we hosted nine visitors from several foreign countries (China, Finland), government agencies (FBI, DoD), and industrial sites (IBM, MITRE, Lincoln Labs). We reviewed the goals of the research program, the architecture of the system, provided several demonstrations, and collected feedback on features and capabilities needed in future versions of the system. Several collaborations resulted from this meeting, including an audio indexing opportunity with George Tech, and invited talks at IBM. More collaborations are planned. At the time this report was written, we are completing plans for the May 2000 workshop, which will include 25 participants, of which 20 are graduate students. Demand for the workshop was strong — we tripled the number of participants over what was originally budgeted. We turned away approximately 10 potential participants due to space and resource limitations.

One surprise in the second year of the project has been the growing importance of supporting the Linux operating system, and the difficulties in doing so. Through experience, we have learned that despite using the same compiler and software (GNU gcc, make, etc.) but a different flavor of Unix (Sun Solaris x86), we cannot guarantee robust releases for Linux users. Hence, we spent some time enhancing our ability to achieve platform independence across a wide range of Unix platforms. We now routinely run our releases through a suite of systems before actually making the release. We also have encountered a large demand for Windows-based ports of our system. Currently, we do this through the use of a Unix-like shell available under Windows (Cygnus' cygwin tools). This has also increased the support overhead in making releases of our system. Despite the demand for a native Windows port, we are not assigning this a high priority until the core system is stable. This is primarily because there is a lack of standardization of C++ compilers and development environments, therefore making it hard to support both environments simultaneously when the software base is changing rapidly.

With the addition of a full-time staff person, it has been possible to expand our on-line support activities. We now handle approximately 5 serious support requests per day. Many support requests involve hand-holding inexperienced users on basic computing issues — we are still struggling with how to deal with these in a timely manner. The remainder involve unexpected program crashes that require extensive diagnosis. We have implemented an automated problem tracking system to make sure such requests are properly tracked. We also provide an ability for users to upload their data to us, so that we can replicate their problems. The majority of serious problems seem to relate to compiler-dependent problems (for example, a bug which did not show up on one version of an operating system, but is fatal on a different architecture with a different compiler). This also exposed the critical need for an in-house multi-platform evaluation facility.

Last but not least, we have made extensive progress enhancing basic functionality of the system. We have developed a generalized hierarchical search engine that is the first such system of its type. We provide an ability to decode speech using either networks for N-gram language models. Users can supply either type of model, or both, at any level of the system. For example, it is possible to constrain the search using N-grams of parts of speech, as well as N-grams of words. We have also developed a front-end that allows users to configure the signal processing portions of the system without writing any code — algorithms can be specified using a GUI-oriented tool that automatically schedules the necessary operations required.

08/15/98 — 08/14/99: 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.