Informal Minicourse: Speech Recognition Tools

Instructor: Mark Hasegawa-Johnson (jhasegaw@uiuc.edu)
TA: Sarah Borys (sborys@uiuc.edu)

May-June, 2005

Existing University of Illinois courses teach speech recognition theory in great depth, but do not teach students how to use the various currently available software packages for training & testing HMMs & SVMs. There is good reason for that: the list of tools changes every year. Nevertheless, students doing current research in ASR eventually want to know how to use current tools. Speech Recognition Tools will attempt to teach you some of the software tools that are useful if you want to design, train, and/or test your own speech recognizers.

1 Lectures, Labs, Grading, and Course Credit

Lectures will be Tuesdays and Thursdays, 1-3 PM, 2369 Beckman.

Originally I did not intend to offer course credit for this course; I intended simply to give lectures, and give away lecture notes and code, to anybody interested in the material. If you would like course credit for this course, however, and if you're a UIUC student, I can probably give you credit for an ECE independent study course; ask if you're interested.

Labs will be distributed weekly. To the extent possible, labs will be designed to be labor-intensive and thorough. They are designed to be pragmatically useful in the short-term rather than theoretically useful in the long term.

If you need help on a lab assignment, please consider asking Sarah (the TA); she's done all of these things in the past, and will help when she knows the answer. Mark is writing the labs, but is likely to respond less quickly than Sarah. If you are working on the lab six months from now, and run into trouble, we will still be happy to help you, time permitting.

Speech data are available on the IFP network, or (if you're at UIUC) we can give you CDs. All code will be available on the web.

2 Lecture Topics

If there is something in particular that you would like to cover, let me know, and I will try to cover it. Things I know well (perl, libSVM, svm_light, HTK) will be covered in most detail. At the time of this writing, I intend to cover the following tools, in the order in which we actually use them in order to design our own recognizers:

1. Recognizer Training
   (a) Data Transcription: Praat; standard speech databases and transcription systems. Dictionaries.
   (b) Transcription conversions, language model training, manipulation of word lattices and recognizer output, and in general, manipulation of any character-string data.
      i. Perl and Python. You must know at least one string-manipulation language that includes hash tables (string-indexed arrays) and regular expressions (RegExes). This course will teach general RegExes and perl. If you already know python or scheme, use that instead. If you use BASIC, awk, or bash, that's good, but you also need to learn perl or python.

ii. SRILM and HTK LM. These are functions for performing common language model and lattice manipulation tasks; we'll review the man pages.

(c) Acoustic Features: HCopy, C and matlab manipulation of speech file formats including SPHERE and HTK.

(d) Phonetic Features: libSVM, SVM-light, PVTK, SPRACH/quicknet, matlab neural nets toolkit. Focus will be on the SVM tools, but we'll also discuss SPRACH/quicknet.

(e) Hidden Markov Models: HTK.

2. Recognizer Testing

(a) Pre-compiled (Static) search space: AT&T and MIT weighted finite state transducer toolkits.

(b) Compile-on-the-fly (Dynamic) search space: HVite implements a token-passing dynamic search space.

(c) Recognition (search): this is actually a current thorn in the side of our research program. HVite is slow. drecog is not open source, so it doesn't easily interface with other tools. In the past, for small static search graphs (small-vocabulary recognition or phone recognition), we have sometimes preferred to write our own decoders in C; we'll discuss what's involved in writing your own decoder.

(d) Off-the-shelf recognizers: given time, we'll delve into the nuts & bolts of Sphinx.