Lecture 1: Transcription and Pre-Processing; Praat and Matlab

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

January 8, 2009

Contents

1 Introduction to the Course

A good paper in speech recognition usually requires four things: (1) a good idea, (2) formal, mathematical proof that your good idea optimizes some criterion of interest, (3) experiments with toy data, under circumstances optimally constructed to showcase the benefit of your idea, (4) experiments with real data, to show that there is some real-world problem that can be nicely solved using your algorithm. In order to prepare for requirement #1, you should read as widely as possible, in your own field, in every related field (information theory, statistics, optimization, graph theory, phonology, history of language, physiology, psychology), and in at least one completely unrelated field of your choice (many people like music; I like poetry and history; Sarah likes animal psychology). In order to prepare for #2 and #3, you should take graduate courses in your area. This "Speech Tools Minicourse" is designed to help with requirement #4.

People who have contributed scripts and sample problems to this course include Sarah Borys, Arthur Kantor, and Bowon Lee; I may add to this list later. Sarah has volunteered to act as TA for this course, which means that you can send her e-mail if you have questions and I'm not available (or if I don't know the answer).

This mini-course exists for you, the graduate student or collaborator who wishes to perform experiments in speech recognition. I will often make unwarranted assumptions about the things that you know or don't know. If anything is unintelligible, please ask.

2 Tools

"Tools" are objects made to help you accomplish a specific task. The open source movement has made it possible to download a great number of tools from the internet. Open source also means that the ultimate documentation for any tool is its source code; this course will teach you, among other things, what to look for in the source code of each tool.

2.1 I need to recognize speech. What should I do?

Here are the steps that I recommend. These are arranged in order of increasing time commitment:

1. Commercial products exist for automatic transcription (speech to text) of English spoken with a north American or British accent using a head-mounted microphone. In my experience, Dragon Naturally Speaking has very low error rate for this task, and individual licenses are inexpensive. If you want to drag and drop waveforms onto a program that will generate text files in response (or dictate live), and if you don't want to change the way it operates in any significant way, I recommend Dragon.

2. If you want a complete end-to-end system, but want more flexibility than Dragon provides, download Sphinx from http://cmusphinx.sourceforge.net/html/cmusphinx.php. Sphinx is a complete open-source ASR distribution, including a decoder, and training tools for acoustic and language models, but also including pre-trained acoustic and language models that work for quiet recording conditions. You can run it out of the box. If it doesn't work well out of the box, it is easy to re-train.

3. If re-training Sphinx doesn't solve your problem, then you need to roll your own speech recognizer, possibly using a wide variety of open-source software tools in an *a la carte* fashion to create the different components. Rolling your own is the subject of this mini-course.

4. If you can't get the recognizer you want by the *a la carte* application of existing tools, then you will need to write your own code, and/or revise the source code of Sphinx and/or HTK and/or other tools described in this course. This course will suggest ways that you can start doing so.

2.2 Tools Developed at Illinois

This course will include some software written at Illinois. Here's how to get it.

- The Cognitive Computation Group publishes packaged software including part of speech taggers, named entity taggers, semantic role labelers, and a number of general machine learning tools at http://l2r.cs.uiuc.edu/ cogcomp/software.php.

- The Statistical Speech Technology Group publishes a letter-to-phone transducer, a voice activity detector, and matlab functions that read and write HTK file formats at http://mickey.ifp.uiuc.edu/wiki/Software.

2.3 Subversion

Some of the Statistical Speech Technology Group software has been gelled but not frozen; in such cases, versions distributed from our subversion repository may be newer than the TGZ archives posted on the web page.

In order to use subversion, you need to do one of these things:

1. On a Windows computer, download and install TortoiseSVN (http://tortoisesvn.net). During the install, tell it which web browser you use. Then, when you click on an SVN link in a web page (e.g., the Mickey Wiki), TortoiseSVN will ask you for login credentials (use username "anonymous" for Mickey Wiki), then show you a complete directory listing. Copy any files you want from the Mickey Wiki to your computer.

2. On a linux computer, a Macintosh, or a Windows computer with Cygwin installed (http://www.cygwin.com), install subversion if it's not already installed (http://subversion.tigris.org), then interact with Mickey Wiki via the command line. This is the method I'll assume for the rest of this section.

In order to get a listing of all of the directories avaiable to you, type

```
svn ls svn://mickey.ifp.uiuc.edu
```

In order to copy a directory to your own computer (to your current working director), e.g., the matlab functions for reading and writing speech files, type

```
svn co svn://mickey.ifp.uiuc.edu/speechfileformats
```

If you've already got a copy on your own computer, change to that directory then update:

```
cd speechfileformats; svn update;
```

If you've made modifications and want to check them in (and have permission to do so),

```
cd speechfileformats; svn ci .;
```

3 Speech Corpora

Speech is free. If one could train a speech recognizer without transcriptions, then the simplest way to get 1000 hours of data would be to put ten tape recorders in front of ten televisions for five days. You wouldn't own the data, and you could not redistribute the data, but you could use it to train your own speech recognizer; training a speech recognizer has thus far been considered to be an example of "fair use" under international copyright law.

Transcription is expensive. Orthographic transcription without time alignment (writing down the words that were spoken in each utterance) can be performed in roughly 4×RT (four hours of transcriber time for each hour of data), and does not require transcribers with special skills, so it costs about $30 per hour of data. Time-aligned orthographic transcription is slightly more expensive. Time-aligned phonetic transcription (write the IPA phone label for each segment) or prosodic transcription (mark pitch accents and phrase boundaries) each require typically 100×RT, and transcribers must be trained for the task, hence they may cost $2000 per hour of data (some have cost more; one corpus reportedly cost $18,000 per hour of data). Expenses of this type require advance planning; in particular, you should probably not use recorded TV broadcasts for this purpose, unless you can somehow secure the legal right to redistribute the audio together with your transcriptions.

As of this writing, I know of four large speech corpora, with transcriptions, that you can download for free from the internet. This minicourse will only use data from these four corpora.

1. AVICAR [8] (http://www.isle.uiuc.edu/AVICAR/) is a database containing seven-microphone, four-camera recordings of speech from one hundred talkers (mostly Illinois students at the time of recording), each reading digits, phone numbers, isolated letters, and phonetically balanced sentences while riding in the passenger seat of a moving car. Any speech researcher can ask for access to the database; access is given via sftp (secure ftp, not regular ftp) to the machine ifp-31.ifp.uiuc.edu.

2. UASpeech [7] (http://www.isle.uiuc.edu/UASpeech/) contains seven-microphone recordings of speech from about eighteen talkers with cerebral palsy, and about eighteen age-matched control subjects (video data have been recorded but not yet segmented for distribution). Each speaker reads 750 isolated words from a 450-word vocabulary. Distribution method is the same as AVICAR.

3. Buckeye (http://www.buckeyecorpus.osu.edu) contains unscripted audio interviews with 137 long-time residents of Columbus, Ohio. This is the only widely available corpus with time-aligned phonetic transcriptions! It is available for free if you sign a license form; the University of Illinois has a copy.

4. AMI Meeting Corpus (Augmented Multiparty Interaction, http://corpus.amiproject.org/) contains recordings of about 40 unscripted or loosely scripted meetings (about 85 hours of data), including head-mounted microphones, tabletop microphones, two microphone arrays, overhead cameras, cameras on the face of each talker, and recordings of the whiteboard over time.

I do not know of any large free speech corpora in other languages. Excellent large corpora are distributed at cost (typically $2k-$10k per corpus) by the institutions that collected them, including the Corpus of Spontaneous Japanese, the Spoken Dutch Corpus, and the European Parliament Corpus (speeches recorded on the floor of the European Parliament, in all the languages of the EU, distributed by ELDA).

The Linguistic Data Consortium (LDC, http://www.ldc.upenn.edu) distributes many nearly-free corpora—"nearly free" in the sense that many of them are free or inexpensive for member institutions, including the University of Illinois. Useful LDC corpora include:

Name	Style	BW	Hrs	Spkrs	Transcriptions
TIMIT	Read	7kHz	14	640	TXT, WRD, PHN
NTIMIT	Read	3.5kHz	14	640	TXT, WRD, PHN
WS97	Conversation	3.5kHz	3.5	2283	TXT, WRD, PHN, Stress, TOBI
Switchboard 1	Conversation	3.5kHz	349	4188	TXT, WRD
Broadcast News	TV	7kHz	Big	Big	TXT
Radio Speech	Read	7kHz	3.5	7	TXT, WRD, TON, BRK, LBL=PHN

1. TIMIT [12]. Transcriptions: WRD (manual word boundary transcription), PHN (manual ARPABET coarse-grained allophonic transcription). SX sentences are "phonetically balanced," i.e., designed so that they cover all of the possible triphone combinations of English. each word). This corpus fits onto 1 CD, is better transcribed than most corpora, is universally cited, and is only $100 from LDC (http://www.ldc.upenn.edu), making it one of the best available bargains in the world of transcribed speech corpora.

2. NTIMIT [6, 9] is the TIMIT corpus passed through real-world telephone channels: low bandwidth, signaling tones, echo, acoustic noise, electrical noise. Sampling rate of the corpus distributed by LDC is still 16kHz, but bandwidth is 3.5kHz (telephone-band), therefore we usually downsample to $F_s = 8$kHz.

3. WS97: Phonetic transcriptions are more detailed than TIMIT. Each phone is transcribed using ARPABET (like TIMIT), then annotated using the most perceptually salient modifier, where modifiers may include manner change (e.g. fricated stops), nasalization, or creak. Prosodic prominence is transcribed using a highly non-standard perceptually based notation [5, 4]. About 200 utterances have also been TOBI-transcribed at UIUC [11]. Because it's conversational speech, coverage of phoneme combinations is MUCH worse than TIMIT. For the same reason, it is quite common to find really dramatic allophonic reduction and assimilation phenomena (e.g., "I don't know" produced as a single nasalized vowel). Transcriptions are available from http://www.icsi.berkeley.edu/real/stp/. Speech data are available by request, from steveng@cogsci.berkeley.edu, for the cost of shipping. Although these speech data are excerpted from the Switchboard 1 corpus, the 1998 Switchboard re-segmentation project [2] eliminated the possibility of time aligning the two corpora.

4. Switchboard [3]. Speech data is distributed in a difficult format (SPHERE two channel format, samples alternating, each sample encoded in 8-bit mu-law). Word boundary time transcriptions are available for free from Mississippi State (http://www.isip.msstate.edu) [2]. Speech data is available from LDC for about $2000, but there are several copies available at UIUC. More recent, larger telephone speech corpora available from LDC include Switchboard 2 (released in 3 sub-corpora, 1999-2001) and the Fisher corpus (about 3000 hours; first subcorpus release was 2005); utterance-level transcriptions are available for these corpora, but not word boundary times.

5. Broadcast News includes announcer speech (read), on-air conversations, on-the-street interviews, and telephone speech, all of which were used for the Hub-4 broadcast news transcription competitions in the late 1990s. Data includes many hours for a few speakers (announcers), and small segments from a large number of speakers (interviewees, etc.). Transcriptions do not include time alignments of the beginning and ending of every word; alignment times are only marked once per "utterance unit."

6. Radio Speech Corpus [10]. Possibly the largest ToBI-transcribed corpus in English; transcriptions include TON (ToBI tone labels) and BRK (ToBI break indices). WRD files contain time-aligned word transcriptions, and POS files contain time-aligned part of speech transcriptions. Also includes automatic phoneme transcriptions (ALA, LBA) and some files have manually aligned phoneme transcriptions (ALN, LBL).

4 Speech File Formats: sox and SPHERE

First, you will have to convert the waveforms into a format that you can understand. In almost all cases, I use primarily the command line tool `sox` to perform format conversions. Here are some of the formats in which speech waveforms may be distributed:

1. SPHERE.
   - Where it is used: LDC distributions, data provided to participants in competitions sponsored by NIST (National Institute of Standards, http://www.nist.gov/speech).
   - File extension: In TIMIT, the files are called WAV, but are actually in SPHERE format; other data in SPHERE format usually has the SPH file extension.
   - Header: A SPHERE header has a length of exactly 1024 bytes, in ASCII (type `man ascii`). In order to read the header of file SX43.SPH, you can just type `more SX43.SPH`.
   - Data: Speech data can be encoded in any number of channels, any number of bytes per sample, any sampling rate, and any encoding scheme: check the file header. TIMIT and NTIMIT are encoded as little-endian short integers, so that they can be easily read using matlab or (on a little-endian architecture, e.g., Intel) using C `fread`. Switchboard is encoded using 8-bit mu-law encoded samples, alternating between channel A and channel B of the conversation.
   - Useful tools: SPHERE 2.6 (http://www.nist.gov/speech) includes C functions for reading and writing SPHERE. If you have the Sphere headers and libraries installed in a place where sox can find them, on a unix system (or a Windows PC with cygwin), then you can just type

     ```
     sox SX43.SPH sx43.wav
     ```

     in order to convert a sphere input file into a Microsoft WAV (RIFF) output file. If you don't have the libraries installed, then 16-bit data (including TIMIT, but not including Switchboard) can be converted in unix using

     ```
     dd bs=1024 skip=1 if=SX43.SPH | sox -t raw -c 1 -r 16000 -w -s - sx43.wav
     ```

     where the `dd` command is only there for the purpose of trimming off the 1024-byte header; the remaining data is treated by sox as if it were raw (-t raw) one-channel (-c 1) 16-bit (-w) signed-integer data (-s) recorded at a sampling rate of 16000 samples/second (-r 16000).

     SPHERE data can be read into matlab using the `readsph.m` function, which you can download as part of the extremely useful voicebox toolkit (http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html).
2. WAV/RIFF. Microsoft WAV format is pretty much an internet standard, and can be read directly into matlab or most other tools. File extension is usually `WAV`.

5 Interactive Analysis and Transcription: Praat

When you put speech data on your computer, the very first thing you should do is to listen to a few files, in order to make sure that you understand the data. For example, this will help you to make sure that you have successfully converted the data into WAV format, without accidentally changing the sampling rate or the number of channels or anything else like that. This will also help you to understand whether the data are noisy or not, how casual vs. formal the data are, what are the dialects and idiolects of the speakers, etc.

In order to interactively analyze and transcribe speech, you should use Praat [1]. Praat is available for every operating system that I've ever used. Download it from www.praat.org.

When you start Praat, you will see two windows: the "Praat objects" window, and the "Praat picture" window. In "objects," choose Read→"Read from file...". Select the audio file you want to load. First, select the audio file you want to view; its name should appear as an entry in the "objects" window.

Now, if you have already created a Praat-format transcription file for this waveform (a TextGrid file), load it using Read→"Read from file...". If not, select the waveform name in "objects," and press the button marked "Annotate"–at the popup, select the "To TextGrid" option. A window will ask you for the names of the transcription tiers you wish. Erase the defaults ("Mary John Bell") and enter the tiers that you want to create (perhaps "words phones speakername"—but if you only want to transcribe words, then replace the

three tiers with the single tier "words"). If any of these tiers will contain instantaneous annotations (labels attached to instants, instead of labels attached to segments), specify these in the box marked "Which of these are point tiers?" When you are done, click "OK," and the name of a new TextGrid file will appear in the "objects" window.

Now select both the waveform and the TextGrid filenames, and choose "Edit." A display will appear that includes the waveform, the spectrogram, and a timeline for each of the transcription tiers. If your waveform is longer than 10 seconds, then the spectrogram display will be empty; you can get a spectrogram display by selecting a short region of the waveform (with the mouse), and choosing View→"Zoom to selection."

Audio playback can be stopped or started by hitting the Tab key; if you have zoomed in on a subset of the waveform, you can play that part by hitting Shift-Tab (if you only want to look at the waveforms, and listen to them, you do not need to create a TextGrid file; just edit the waveform by itself).

Praat automatically draws blue vertical lines at the detected glottal closure instants. In order to overlay a pitch track on top of the spectrogram, choose Pitch→"Show Pitch." Pitch→"Pitch Settings" allows you to adjust the pitch settings; for example, if you know the lowest and highest pitch frequencies used by a particular talker, then you can improve the accuracy of the algorithm by constraining it to consider only pitch frequencies within the specified range. Pitch→"Pitch listing" creates a text file with pitch frequencies measured once every 10ms throughout the file.

To transcribe the waveform, select a segment of the audio file, then hit Ctrl-1 to mark that segment on Tier 1 (Ctrl-2 for tier 2, etc.). In order to enter a label for the newly created segment, just click on it (in tier 1), and type the desired label. When you are done, save the TextGrid (if you wish) using File→"TextGrid to text file...".

Praat also has an excellent scripting facility, which is really useful if you want to perform the same analysis on many different waveforms. I have not personally used Praat scripting; if a task becomes that complicated, I use matlab.

6 Transcription File Formats

As noted earlier, most speech databases are distributed with transcriptions. ALL important transcription formats are formatted plaintext, therefore:

1. If you are unsure what type of file you are dealing with (or what sub-format it has, in the case of SPHERE files), use `more` or `less` to look at it.

2. Once you know the file format, you can easily reformat it to a different file format using your favorite scripting language (e.g., perl, python, or ruby).

Perl scripts for converting among a few of the common speech transcription file formats are available at svn://mickey.ifp.uiuc.edu/transcription_tools. For example, if you wish to convert from a TIMIT transcription into a TextGrid (so that you can view the TIMIT transcription more easily in Praat—but this is not actually necessary, I think, because Praat can read TIMIT transcriptions natively), you should be able to type

```
echo SX43.PHN > test.scp
sphere2mlf.pl test.scp 16000 phn > SX43.mlf
mlf2praat.pl test.scp SX43.mlf > SX43.TextGrid
```

Documentation for each of these perl scripts is available by calling the filename with no arguments, e.g., calling `sphere2mlf.pl` will bring up information about the program.

Typical transcription file formats include:

1. SPHERE/NIST

   - Where it is used: LDC distributions, NIST competitions
   - File Extension: Variable. Utterance-level segmentation is contained in .TXT (TIMIT) or × -trans.text (Switchboard) files. Word-level segmentation is contained in .WRD (TIMIT) or × -word.text (Switchboard). Phone-level segmentation may be in PHN files.
   - Header: usually none.

- Data: SPHERE format transcriptions consist of one segment per line. Each line matches the following pattern:

  `[CORPUS\_NAME] START\_TIME END\_TIME LABEL1 [LABEL2 ...] \# [COMMENT]`

  CORPUS_NAME is optional; if present, it must be a string containing no spaces. START_TIME and END_TIME may be written in samples (TIMIT, NTIMIT: at 16kHz sampling rate), or in seconds (Switchboard). There may be many sequential labels per segment, e.g., a TXT file includes the entire sentence in a single line. Hash marks the beginning of a comment.

2. ESPS (Entropic Signal Processing Systems)

   - Where it is used: Radio Speech Corpus.
   - File Extension: Variable. Radio Speech uses .WRD, .TON, .BRK, .LBA, and .LBL.
   - Header: specifies the number of fields (`nfields`). Ends in # on a line by itself.
   - Data: Each line specifies a point in time. Usually, it is assumed that the end time of one segment is equal to the start time of the next segment. Format of each line:

     `START\_TIME COLOR LABEL1 [LABEL2 ...]`

     START_TIME is always in seconds. COLOR is an integer, specifying the colormap entry of the color used to label this particular time alignment. The purpose of COLOR is to allow color-coded display of multiple transcriptions at the same time; to my knowledge, only the original WaveSurfer makes use of this detail.

3. MLF (Master Label File).

   - Where it is used: HTK (the hidden Markov modeling toolkit) usually requires all other transcriptions to be converted into this format before training or testing a recognizer. MLF is convenient because it puts transcriptions for many speech files into a single transcription file.
   - File Extension: MLF.
   - Header: HTK will reject an MLF file not following this format: The file header is one line containing the characters "`#!MLF!#`". The section of the file corresponding to one speech file is initiated by a line naming the file, and is ended by a line containing a period (`.`) on a line by itself. In theory, the filename line is relatively flexible, but in practice, HTK breaks unless it contains exactly the characters

     `"*/FILENAME.lab"`

     where FILENAME is replaced by the root name of the speech file (everything that is not part of the path or the extension).
   - Format of a data line is

     `START\_TIME END\_TIME LABEL1 [LABEL2 ...]`

     START_TIME and END_TIME are optional; if the first non-space character on the line is not an integer, HTK will assume that the start time and end time of this line are not specified. If START_TIME and END_TIME are present, they must be integer values, specified in 100ns units (thus the line "450000 480000 DX" specifies a flap exactly 30ms long, starting at 0.45 seconds, and continuing until 0.48 seconds).

4. TextGrid.

   - Where it is used: this is the file format for Praat transcriptions.
   - File Extension: always TextGrid (e.g., SX43_PHN.TextGrid).

- Header, Data formats: These are too complicated to describe here, but relatively easy to learn if you need to. TextGrid is the only format listed here that can contain many different non-synchronous transcriptions of the same speech waveform in a single transcription file. The data format is designed using envelopes, rather like HTML, but with a different syntax. The outermost envelope is the file; then the transcription tier; then the segment or point. Within the segment envelope are contained fields (separate lines) specifying start time, end time, and label. The point envelope is similar to the segment envelope, but specifies just one time, rather than two times.

5. SGML
   - Where it is used: Broadcast News
   - File Extension: SGML or XML.
   - Header: one line containing the filename, e.g. "m970927.sgml".
   - Data: in standard SGML envelopes, with types like "episode" (wrapping a broadcast program), "section" (wrapping a section with a particular content and a particular channel quality), "turn" (wrapping the dialog turn of one speaker), and "time" (specifying an alignment time at which the speech file may be broken; the "time" envelope is usually not closed by a /time marker).

7 Denoising and Voice Activity Detection: matlab

Noise and reverberation are a really big deal; even with the best compensation techniques, they can multiply the error rate of ASR by tenfold. I recommend the following techniques:

1. If a waveform is corrupted by an obvious deterministic signal, e.g., a 60Hz hum, then get rid of the hum by estimating it and then subtracting it directly from the waveform.
2. If a waveform is corrupted by audible noise, correct it using spectral subtraction. SS works best with stationary noise (fan noise, wind noise, motor noise), and less well with non-stationary noise (music, background speech).
3. Perform voice activity detection, and throw away the silences.
4. Convert to the cepstral/spectral representation that you will be using for ASR (we'll talk about these in the next lecture).
5. Perform cepstral mean subtraction, variance normalization, and ARMA filtering, using the MVA tool from http://ssli.ee.washington.edu/people/chiaping/mva.html. "Mean subtraction and variance normalization" is exactly the same thing as z-normalization in statistics, i.e., for each feature x_t (at time t) that has a mean of μ and a standard deviation of σ, we compute

   $$z = \frac{x - \mu}{\sigma}$$

   the only trick, here, is that μ and σ should be the averages computed over all speech files that contain speech by the same talker, in the same channel (same room, same microphone, same background noise). The more speech you have that meets this criterion, the better; e.g., it's best to use more than one speech file to computer μ and σ if you can.
6. Finally, and MOST EFFECTIVE: train your acoustic models using speech that has been recorded and processed, to the extent possible, in exactly the same way as your test data (same noise environment, same room, same microphone, et cetera).

Of the steps listed above, steps #1, #2, and #3 can only currently be performed in matlab. If you don't have matlab on your computer, consider installing it: it can be purchased for low cost by Illinois students. The open source program `octave` (http://www.gnu.org/software/octave) is supposed to have similar basic signal processing functionality, but I haven't tested it yet, myself.

In addition to a basic matlab installation, you want to get:

1. The voicebox toolkit (http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html), and
2. The speechfileformats and lee_vad packages from mickey (svn://mickey.ifp.uiuc.edu/lee_vad, svn://mickey.ifp.uiuc.ed

Put these tools in your home directory (e.g., I have mine in /home/jhasegaw/matlab/speechfileformats), then add them to your matlab path.

7.1 Learn Matlab

The first thing you will want to do is to learn matlab. I recommend the MIT tutorial "MATLAB on Athena," http://web.mit.edu/olh/Matlab/Matlab.html.

7.2 Remove Deterministic Hum

This step is necessary only if your waveform is corrupted by a deterministic hum, e.g., by 60Hz electrical line noise. A reasonably effective way to get rid of 60Hz line noise is as follows:

1. Start matlab.
2. Read the waveform: `[x,fs]=readwav('sx43.wav');`—readwav is provided in the voicebox toolkit, but matlab also comes distributed with a function called wavread that does the same thing.
3. Compute the Fourier transform of the entire waveform: `X=fft(x);`
4. Compute a frequency axis for your fft plot: `F=[0:(length(x)-1)]*(fs/length(x));`
5. Plot the long-term magnitude-squared FFT: `plot(F,abs(X).^2);`
6. If your waveform is being corrupted by 60Hz line noise, then you should see sharp peaks at 60Hz, 120Hz, and so on. You can zero out those peaks, and the spectral samples immediately to left and right of each peak, using something like this:

   ```
   badfreqs=[60:60:(fs/2)];
   bad0=[badfreqs, fs-badfreqs]*(length(X)/fs)-1;
   bad1=badindices1-1;
   bad2=badindices1+1;
   X([bad0,bad1,bad2])=zeros(size([bad0,bad1,bad2]));
   ```

   Try plotting `abs(X).^2`. If the peaks are still too high, try zeroing out bad0-2 and bad0+2, and so on.
7. Now try `x2=ifft(X);`, and plot the waveform `x2`, to make sure that the hum is gone, and that the speech is not distorted.

Another method to do the same thing is to use a comb filter, e.g.,

```
x2=conv(x,[1 zeros(1,round(fs/60)-2),-0.99]);
```

I think that comb filters are also available in Praat.

7.3 Spectral Subtraction

If there is audible wind noise or fan noise in the background, it usually helps to correct it with spectral subtraction. The buzz considered in the last section was perfectly predictable, and therefore it could be subtracted exactly. Wind noise is not perfectly predictable: we know the average noise power in each frequency bin, therefore we can correct for the *average* noise power—but that's not the same as correting for the *true* noise power at any given instant. All the same, it's much better than nothing; in fact, many experiments find that spectral subtraction is better than a lot of more sophisticated techniques for ASR purposes.

Basic algorithm, in matlab, is something like this:

1. Start matlab.
2. Read the waveform: `[x,fs]=readwav('sx43.wav');`
3. Choose a frame skip. 10ms is common in speech recognition, but certainly not necessary: `SKIP=round(0.01*fs);`
4. Create a Hanning window of length equal to exactly twice the frame skip. If you have the signal processing toolkit installed, you can type `window=hanning(2*SKIP);`. Otherwise, try

   ```
   window=0.5-0.5*cos(pi*[0:(2*SKIP-1)]/SKIP);
   ```
5. Chop the waveform up into frames, using the `enframe` function provided in the voicebox toolkit:

   ```
   help enframe;
   f=enframe(x,window,SKIP);
   ```
6. Fourier transform the frames. The `enframe` function puts frames into the rows of `f`, but `fft` transforms the columns of its argument, so you want to type

   ```
   F=fft(f');
   ```

   where the single-quote computes the transpose of the `f` matrix.

   ```
   [NFREQS,NFRAMES]=size(f);
   ```

   The last command finds out how many frames there are (one frame per column of the matrix `F`) and how many different frequencies there are (one frequency bin per row of the matrix `F`). `NFREQS` should be the same as `2*SKIP`, if everything has worked properly so far.
7. Compute the power and phase of `F`. Each of these should be an NFREQS×NFRAMES matrix.

   ```
   FPOWER=abs(F).^2; FPHASE=angle(F);
   ```
8. Estimate the average noise power spectrum. Suppose that you are reasonably sure that the first twenty frames (the first 200ms) contain only noise, with no speech. Then you can type

   ```
   NPOWER=mean(FPOWER(:,1:20)')';
   ```

   Note that the 1:20 selects out the first 20 frames of `FPOWER`, creating a matrix with only 20 columns (one column per frame). That matrix is transposed, so that we have 20 rows (one row per frame). The `mean` function in matlab computes the average of each column—that is, the average power at each frequency, averaged across the first 20 frames. The result is a row vector. The last transpose operation converts the row vector back to a column vector.
9. Repeat the average noise power once for each frame in the utterance:

   ```
   NPOWER_REP=repmat(NPOWER,1,NFRAMES);
   ```

   and subtract it from the signal power, clipping at zero:

   ```
   GPOWER=max(0,FPOWER-NPOWER_REP);
   ```
10. Create a denoised FFT by taking the square root of the denoised power spectrum, times the original (possibly noisy) phase. Rationale: in the frequency bins where speech is louder than noise, `FPHASE` is approximately equal to the true speech phase. In frequency bins where noise is louder than speech, `GPOWER` is almost zero, so its phase doesn't matter.

    ```
    G=sqrt(GPOWER).*exp(j*FPHASE);
    ```
11. Inverse FFT, and transpose:

    ```
    g=ifft(G)';
    ```
12. Now, create two estimated waveforms. The first one, `xhat1`, is just a check, to make sure that enframing and deframing works correctly; it is created from the original enframed signal, therefore it should be identical to the original signal. The second one, `xhat2`, is the signal with average noise spectrum removed:

```
xhat1=zeros(1,(NFRAMES-1)*SKIP+length(window);)
xhat2=zeros(1,(NFRAMES-1)*SKIP+length(window);)
for FRAMENUM=1:NFRAMES,
  xhat1((FRAMENUM-1)*SKIP+[1:length(window)]) += f(FRAMENUM,:);
  xhat2((FRAMENUM-1)*SKIP+[1:length(window)]) += g(FRAMENUM,:);
end
```

13. Plot the waveforms, to make sure that everything worked. Code below plots `x` in blue, `xhat1` in black, and `xhat2` in red.

```
tx=[1:length(x)]/fs;
that=[1:length(xhat1)]/fs;
plot(tx,x,'b',that,xhat1,'k',that,xhat2,'r');
```

14. Finally, write out the new waveform:

```
writewav(xhat2,fs,'sx43_denoised.wav');
```

7.4 Voice Activity Detection

After you've performed spectral subtraction, it often helps to trim away the long silences. By doing so, you can keep the speech recognizer from mis-recognizing, as speech, some burst of wind noise.

A matlab function for voice activity detection is available at svn://mickey.ifp.uiuc.edu/lee_vad. Add it to your matlab path, then type `help VAD;` for more information.

References

[1] Paul Boersma and David Weenink. Praat: doing phonetics by computer. Technical report, Institute of Phonetic Sciences, University of Amsterdam, 2003.

[2] Neeraj Deshmukh, Aravind Ganapathiraju, Andi Gleeson, Jonathan Hamaker, and Joseph Picone. Resegmentation of switchboard. In *Proc. Internat. Conf. Spoken Language Processing*, pages 0685:1–4, 1998.

[3] J.J. Godfrey, E.C. Holliman, and J. McDaniel. SWITCHBOARD: telephone speech corpus for research and development. In *Proc. ICASSP*, pages 517–520, San Francisco, 1992.

[4] S. Greenberg, H.M. Carvey, and L. Hitchcock. The relation of stress accent to pronunciation variation in spontaneous american english discourse. In *Proc. ISCA Workshop on Prosody and Speech Processing*, 2002.

[5] Steven Greenberg and Leah Hitchcock. Stress-accent and vowel quality in the Switchboard corpus. In *NIST Large Vocabulary Continuous Speech Recognition Workshop*, Linthicum Heights, MD, May 2001.

[6] Charles Jankowski, Ashok Kalyanswamy, Sara Basson, and Judith Spitz. NTIMIT: A phonetically balanced, continuous speech, telephone bandwidth speech database. In *Proc. ICASSP*, 1990.

[7] Heejin Kim, Mark Hasegawa-Johnson, Adrienne Perlman, Jon Gunderson, Thomas Huang, Kenneth Watkin, and Simone Frame. Dysarthric speech database for universal access research. In *Proc. Interspeech*, 2008.

[8] Bowon Lee, Mark Hasegawa-Johnson, Camille Goudeseune, Suketu Kamdar, Sarah Borys, Ming Liu, and Thomas Huang. AVICAR: Audio-visual speech corpus in a car environment. In *Proc. Interspeech*, pages 2489–92, Jeju Island, Korea, 2004.

[9] Pedro J. Moreno and Richard M. Stern. Sources of degradation of speech recognition in the telephone network. In *Proc. ICASSP*, volume I, pages 109–112, 1994.

[10] Mari Ostendorf, Patti J. Price, and Stefanie Shattuck-Hufnagel. The boston university radio news corpus. Technical Report LDC96S36, Linguistic Data Consortium, 1995.

[11] Taejin Yoon, Sandra Chavarria, Jennifer Cole, and Mark Hasegawa-Johnson. Intertranscriber reliability of prosodic labeling on telephone conversation using ToBI. In *Proc. Interspeech*, pages 2729–32, Jeju Island, Korea, 2004.

[12] V.W. Zue, S. Seneff, and J. Glass. Speech database development at MIT: TIMIT and beyond. *Speech Communication*, 9:351–356, 1990.