EE627 – Term Project : Jul. 2013 Semester

August 13, 2013

Title : Build and demonstrate a real time continuous speech recognition system in Hindi

Assigned to : Batch No. 5

TAs Assigned : Lalan. He is Available @ACES 203 MiPS Lab EE Department. email : lalank@iitk.ac.in

Objective :

The objective of this project is to build a continous speech recognition system for Hindi. A real time speech recognitiion needs to be built using the District and Crop database. Experimental results for speech recognition in terms of word error rate (WER) also need to be provided.

Methodology To Be Followed :

The methodlolgy need to be followed for performing the recogniton experiments are as follows:

- Building the task grammar (a "language model")
- Constructing a dictionary for the models
- Creating transcription files for training data .
- Encoding the data (feature processing)
- (Re-)training the acoustic models
- Evaluating the recognizer against the test data
- Reporting recognition results
- Real time system development

Database for Training

The database used for training is District and Crop database. The district and crop database are obtained by recording the districts names and crop names from several speakers. Majority of the speaker population were students and employees of various departments and centers of Indian Institute of Technology Kanpur. Speakers were selected based on parameters such as diversity of age, dialect and gender as these parameters are very important in the effective design of a database. The distribution of the male female ratio was the same in developing the audio archives.

Tools To Be Used

The Tools to be used to implement this term project is the Sphinx Toolkit. The details of download, installation, and usage are available at the following URL :

http://cmusphinx.sourceforge.net/

Deliverables/ Submissions

The deliverables or submission procedures for the term project are as follows:

- Presentations and Report : Two set of presentations is required for every batch in this term project. The first presentation will be scheduled before mid sem and the second presentation will be scheduled before end sem. The marks will be distributed separately for two presentation. Additionally a report needs to be submitted detailing the project.
- System Demo in real time : The demonstration of the project is need to be carried out by each batch. The demonstration includes the real time presentation of the working model for recognition system.
- Code/script submissions : The code or script has to be submitted by each batch which will include complete details of the project.

Other Useful Links

The other useful links that might be helpful in preparation of the code or script are as follows

- SPHINX TOOLKIT : http://www.cs.cmu.edu/~archan/sphinxInfo.html
- Matlab Code : https://www.mathworks.in/matlabcentral/linkexchange/links/3703-matlab-sphinx-documentation

PHONE MODEL USING SPHINX3-0.8

****
INSTALLING SPHINX ON A GNU/LINUX SYSTEM
****

1. Goto Home directory and Create a folder PhoneModel in the home directory.

```
mkdir PhoneModel
```

2. Create a directory called workspace in PhoneModel

```
mkdir workspace
```

3. In the workspace directory make 3 directories namely tools,source and downloads using the command

```
mkdir downloads tools source
```

4. Copy all the zip files to the downloads directory

a. sphinx3-0.8.tar.gz
b. sphinxbase-0.6.tar.gz
c. SphinxTrain-1.0.tar.bz2

5. Untar the zip files sphinx3-0.8.tar.gz sphinxbase-0.6.tar.gz using commands

```
tar -xvzf sphinx3-0.8.tar.gz
tar -xvzf sphinxbase-0.6.tar.gz
```

6. Move sphinx3-0.8 and sphinxbase-0.6 folders to source directory using command

```
mv sphinx3-0.8 ../source
mv sphinxbase-0.6 ../source
```

7. Also make 3 directories in workspace/tools directory using the command

```
mkdir sphinx3 sphinxbase sphinxtrain
```

8. In the path workspace/source/sphinxbase-0.6 run the following command to compile and install sphinxbase-0.6

```
./configure --prefix=/home/PhoneModel/workspace/tools/sphinxbase
make
make install
```

This configures, compiles and installs the sphinxbase.The above command creates 3 folders in tools/sphinxbase i.e., bin, include and lib,of which there are executable binary files in bin folder. 2

files that are mainly used sphinx_fe and sphinx_jsgf2fsg

9. In the path workspace/source/sphinx3-0.8 run the following command to compile and install sphinx3-0.8

```
./configure --prefix=/home/PhoneModel/workspace/tools/sphinx3 --with-sphinxbase=/home/PhoneModel/workspace/source/sphinxbase-0.6
make
make install
```

This will configure, compile and install the sphinx3.The above command creates 4 folders in tools/sphinx3 i.e., bin include lib and share,of which there are executable binary files in bin folder.One of the files that is mainly used that is sphinx3_decode

10. In the downloads directory bunzip the SphinxTrain-1.0.tar.bz2 file to get SphinxTrain-1.0.The command used is

```
bzip2 -cd SphinxTrain-1.0.tar.bz2 | tar -xv
```

11. move SphinxTrain-1.0 to the sphinxtrain directory inside tools directory

```
mv SphinxTrain-1.0 ../tools/sphinxtrain
```

12. In the path tools/sphinxtrain/SphinxTrain-1.0 execute the following commands

```
./configure
make
```

This configures and installs the Sphinxtrain.

install -- cmuclmtk -- move cmuclmtk in worspace/source. create cmuclmtk directory in workspace/tools from workspace/source/cmuclmtk/--

14. in the etc/skel directory set the following environment variables by executing the following command

./configure --prefix=/home/lalan/PhoneModel/workspace/tools/cmuclmtk (note--sometime with prefix, tab does not work, so give location....)
make
make install

```
$ vim ~/.bashrc
```

at the end of the file add the following environmental variables

```
export SPHINXTRAIN=/home/PhoneModel/workspace/tools/sphinxtrain/SphinxTrain-1.0
export SPHINXDIR=/home/PhoneModel/workspace/tools/sphinx3
export SPHINXBASE=/home/PhoneModel/workspace/tools/sphinxbase
export CMULM=/home/PhoneModel/workspace/tools/cmuclmtk/
```

save and exit by

```
Esc
:wq
```

next execute the command

```
$ source ~/.bashrc
```

close the terminal and re-open a new terminal

****
DATA PREPARATAION
****

15.Save all your wave files with .wav extension

16.Create a list of all the wave files as shown in calflow_train.fileids. For example if the wave files are Abc.wav,Bcd.wav and Cde.wav, then the list will have

```
Abc
Bcd
Cde
```

as enteries. Save this as taskword_train.fileids.(The taskword can be any name of your choice. In our case it is calflow1)

17.Create a list of transcription of all the wave files as shown in calflow1_train.transcription. For example if the contents of the wav files are xxyyzz, ppqqrr, uuvvww respectively. then the transcription will have

```
<s> xxyyzz </s> (Abc)
<s> ppqqrr </s> (Bcd)
<s> uuvvww </s> (Cde)
```

as enteries. Save this as taskword_train.transcription. Note maintain equal spacing between all the words in all the lines.

18.Create a filler dictionary with all non-speech sounds with the name taskword.filler as shown by calflow.filler. In our case silence is the only non-speech sound being used hence the enteries of the filler dictionary are

```
<s>     SIL
</s>    SIL
<sil>   SIL
```

19. Create a dictionary of all the words that are there in the vocabulary of the recogniser and save it as taskword.dic as shown by calflow.dic. Here the enteries of the dictionary will be

```
xxyyzz        xx yy zz
ppqqrr        pp qq rr
uuvvww        uu vv ww
```

again make sure that the spacings are regular in each line and there are no extra blank lines in the end

20. Create a list of all the phones that are being used and save it as taskword.phone as shown by calflow.phone. Here the enteries of the phone list will be

Note that the phone list is the sorted version n not--just shown here.

```
xx
yy
```

zz
pp
qq
rr
uu
vv
ww
SIL

make sure that the SIL is included in the phone list

****
SPHINX TRAIN
****

21. In the workspace directory create the directory hmm using the command

```
mkdir hmm
```

22. Goto the hmm directory and execute the following commamnd

```
cd hmm
```

this dollar sud be there aprt from terminal $.

```
$SPHINXTRAIN/scripts_pl/setup SphinxTrain.pl -task taskword
```

The taskword can be chosen by the user as per convenience, in our case it is calflow1.
The above command sets up all the folders and files required for training.
The above script generates the following important directories in hmm directory.

a. etc/ : which contains the configuration files and the required transcript and dictionary files. Move taskword.dic, taskword.filler. taskword.phone, taskword_train.fileid and taskword_train.transcription to this folder. It already contains two files named feat.params and sphinx_train.cfg. The contents of feat.params should be as follows which set the parameters for feature extraction

```
-alpha 0.97
-srate 8000
-frate 100
-dither yes
-doublebw no
-nfilt 31
-ncep 13
-lowerf 200
-upperf 3500
-nfft 512
-wlen 0.0256
-transform legacy
-feat __CFG_FEATURE__
-agc __CFG_AGC__
-cmn __CFG_CMN__
-varnorm __CFG_VARNORM__
```

The contents of sphinx_train.cfg give the parameters to be passed during training. The following modifications are required to be made in this file

> $CFG_WAVFILE_EXTENSION = 'sph' to be changed to 'wav' ; that is to say, the file extension is .wav
> $CFG_WAVFILE_TYPE = 'nist' has to be changed to 'mswav' ; that is Microsoft Wave file

The number of HMM and GMMS can also be changed as per requirement.

b. wav/ : which contains raw audio data for training the models. Move all your wave files to this folder

c. feats/ : which contains the mfc files generated from the raw audio files after feature extraction.

d. model_parameters/ : which contains our final models (means, mixture weights, transition matrices and variances) obtained after the training process.

e model_archtecture/ : which contains the model definition files and topology of basic HMM

23. go to the hmm dir and execute the command

```
perl scripts_pl/make_feats.pl -ctl etc/taskword_train.fileids
```

The above command extracts the feature in the form of .mfc files and saves them in the feat directory

24. Then run the command

```
$ perl scripts_pl/RunAll.pl
```

in message, u may find many error and warning messages and process continues, but no need to worry, worry when process stops. step completion u can see in hmm folder task.html

The whole process may take some time. This creates the trained vectors which will be used while decoding.

Note: For force allignment move the sphin3_decode binary file which is located at tools/sphinx3/bin to hmm/bin. Enable the force allignment configuration part in sphinx_train.cfg.

Then retarian the system after making the following changes in RunAll.pl located at hmm/scripts_pl

```
 ("$ST::CFG_SCRIPT_DIR/00.verify/verify_all.pl",
# "$ST::CFG_SCRIPT_DIR/01.vector_quantize/slave.VQ.pl",
"$ST::CFG_SCRIPT_DIR/02.falign_ci_hmm/slave_convg.pl",
"$ST::CFG_SCRIPT_DIR/03.force_align/slave_align.pl",
# "$ST::CFG_SCRIPT_DIR/04.vtln_align/slave_align.pl",
#"$ST::CFG_SCRIPT_DIR/05.lda_train/slave_lda.pl",
#"$ST::CFG_SCRIPT_DIR/06.mllt_train/slave_mllt.pl",
#"$ST::CFG_SCRIPT_DIR/20.ci_hmm/slave_convg.pl",
# "$ST::CFG_SCRIPT_DIR/30.cd_hmm_untied/slave_convg.pl",
        #"$ST::CFG_SCRIPT_DIR/40.buildtrees/slave.treebuilder.pl",
        # "$ST::CFG_SCRIPT_DIR/45.prunetree/slave.state-tying.pl",
        #"$ST::CFG_SCRIPT_DIR/50.cd_hmm_tied/slave_convg.pl",
        #"$ST::CFG_SCRIPT_DIR/90.deleted_interpolation/deleted_interpolation.pl",
        #"$ST::CFG_SCRIPT_DIR/99.make_s2_models/make_s2_models.pl",
   );
```

This will create a directory called faligmout inside hmm dirctory. Copy the files taskword.alignedfiles and taskword.alignedtranscripts.1 present in falignout directory and replace taskword_train.fileids and taskword_train.transcription respectively with these files.
Once this done undo the changes done in RunAll.pl and sphinx_train.cfg and then retarin the sytem to obtain the final models.

****
SPHINX DECODE
****

25. Create the decode directory in the path /home/PhoneModel/workspace/

```
mkdir decode
```

26. In the path /home/PhoneModel/workspace/decode create the subfolders by name feats, models and wav folders where,

   a.) feats directory contains all the .mfc (feature extracted files) files
   b.) wav   directory contains all the test wav files

27. In the path /home/PhoneModel/workspace/decode/models create the subfolders hmm and lm

   a.) hmm   directory has the following files created during training: feat.params, mdef, means, mixture_weights, transition_matrices, variances

   b.) lm   directory has the following files: taskword.dic, taskword.filler and the taskword.fsg files

All the above files in hmm and lm are copied from the trained models executing the following commands in the path /home/PhoneModel/workspace/

```
cp hmm/model_parameters/taskword.cd_cont_1000_8/* decode/models/hmm/

cp hmm/model_architecture/taskword .1000.mdef decode/models/hmm/mdef

cp hmm/etc/taskword.dic ../decode/models/lm

cp hmm/etc/taskword.filler ../decode/models/lm

cp hmm/etc/feat.params ../decode/models/hmm
```

this dots & / not required

28. Make taskword.jfsg file in the lm directory wich has the following format

```
#JSGF V1.0;

grammar topping;
```

```
public <topping> = ( SIL* ( [<words>] ) SIL*);

<words> = (xxyyzz | ppqqrr | uuvvww):
```

now run the following command to create the taskword.sfg file from lm directory in decode/mode ls/lm

```
 /home/PhoneModel/workspace/tools/sphinxbase/bin/sphinx_jsgf2fsg assamese2.jsfg >
assamese2.fsg
```

29. the feat.params should have the following enteries in decode/model/hmm

all parameter same as during traning except last four parameters are not there.

```
-alpha 0.97
-samprate 8000
-frate 100
-dither yes
-doublebw no
-nfilt 31
-ncep 13
-lowerf 200
-upperf 3500
-nfft 512
-wlen 0.0256
```

30.Feature Extraction Command for Testing is

```
/home/PhoneModel/workspace/tools/sphinxbase-0.6/bin/sphinx_fe -argfile
models/hmm/feat.params -c test_files -di wav/ -do feats -ei wav -mswav yes -eo mfc
```

Where
-argfile is specified as models/hmm/feat.params.
-mswav specifies the Microsoft Wave file format.
test_files contains the list of all the wave files used for testing prepared is a manner similar to that of training

31.the command for decoding is as follows

```
/home/PhoneModel/workspace/tools/sphinx3-0.8/bin/sphinx3_decode -hmm
models/hmm -op_mode 2 -fsg models/lm/assamese2.fsg -dict models/lm/assamese2.dic
-fdict models/lm/assamese2.filler -ctl test_files -logfn log.txt -hyp out.txt -cepdir feats/
```

Where,

-hmm	Directory for specifying Sphinx 3's hmm, the following files are assummed to be present, mdef, mean, var, mixw, tmat. If -mdef, -mean, -var, -mixw or -tmat are specified, they will override this command.
-op_mode	Operation mode, for internal use only. Since FSG is the mode used so -op_mode has to be set to 2

-fsg	Finite state grammar.
-ctl	Control file listing utterances to be processed (List of file that has to processed)
-logfn	Log file (log.txt)
-hyp	Recognition result file, with only words (out.txt)
-cepdir	Input cepstrum files directory (prefixed to filespecs in control file) (Where, feats/ directory contain all the test mfc files)
out.txt	has the list of files recognised and their corresponding words

****
WORD ERROR RATE
****

Once decoding is sucessful then to compute the word error rate following command is needs to executed

```
perl /home/PhoneModel/workspace/hmm/scripts_pl/decode/word_align.pl ReferenceOut.txt
out.txt > temp
```

Where,

ReferenceOut.txt	Reference Output file
out.txt	is the output file generated using the decoding step
temp	stores the individual and overall accuracy scores

Note: To generate the timing duration or frame durations of voiced and unvoiced regions in the wave files goto /source/sphinx3-0.8/src/tests/regression directory and create a shell script as shown

```
#!/bin/sh

echo "ALIGN TEST simple"

tmpout="test-align-simple.seg"

#Simple test
sh ../../../libtool --mode=execute ../../../src/programs/sphinx3_align -logbase 1.0003 -mdef
/home/PhoneModel/workspace/hmm/model_parameters/calflow1.ci_cont/mdef -mean
/home/PhoneModel/workspace/hmm/model_parameters/calflow1.ci_cont_32/means -var
/home/PhoneModel/workspace/hmm/model_parameters/calflow1.ci_cont_32/variances -mixw
/home/PhoneModel/workspace/hmm/model_parameters/calflow1.ci_cont_32/mixture_weights
-tmat
/home/PhoneModel/workspace/hmm/model_parameters/calflow1.ci_cont_32/transition_matrices
-feat 1s_c_d_dd -topn 1000 -beam 1e-80 -senmgau .s3cont. -agc max -fdict
/home/PhoneModel/workspace/hmm/etc/calflow1.filler -dict
/home/PhoneModel/workspace/hmm/etc/calflow1.dic -ctl
```

```
/home/PhoneModel/workspace/hmm/etc/calflow1_train.fileids -cepdir
/home/PhoneModel/workspace/hmm/feat/ -insent
/home/PhoneModel/workspace/hmm/falignout/calflow1.aligninput -outsent $tmpout -wdsegdir ./
-phsegdir ./ > test-align-simple.out 2>&1
```

This will create files that will contain the word level as well as phone level time duration for each wavefile

Learn decoding using Sphinx III

March 28, 2011

Pranav Jawale, DAPLAB, IIT Bombay

After following this tutorial you should be able to

Given a set of audio files, create dictionary, language model etc. and run the Spinx3 decoder to get output of speech recognizer. Draw inferences from the log files that are created as a result. **The reader is assumed to be working on a MS Windows machine.**

Mother website of Sphinx: http://cmusphinx.sourceforge.net/

1. http://cmusphinx.sourceforge.net/wiki/ [Collaborative documentation]
2. http://cmusphinx.sourceforge.net/wiki/research/ [List of publications]
3. [**User forums**]
   Speech Recognition - Generic discussions about speech recognition
   Sphinx3 Sightings - News and announcements about sphinx3
   cmusphinx-devel - For contacts of activities of development of all Sphinx components
   Feature Requests - Feature Request Tracking System
4. [**The Hieroglyphs**: Building Speech Applications Using CMU Sphinx and Related Resources – by various Sphinx developers] Original link: http://www.cs.cmu.edu/~archan/sphinxDoc.html
   Local link: http://home.iitb.ac.in/~pranavj/daplabwork/hieroglyph_sphinx.pdf
5. Some more links http://www.cs.cmu.edu/~archan/sphinxInfo.html
6. **Decoder description [must read]**
   http://www.cs.cmu.edu/~archan/s_info/Sphinx3/doc/s3_description.html
7. **The CMU-Cambridge Statistical Language Modeling Toolkit**
   http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html
8. Homeworks in **Speech Processing -- Fall 2010** http://www.speech.cs.cmu.edu/15-492/
9. **Speech recognition seminars** at Leiden Institute for Advanced Computer Science, Netherlands
   http://www.liacs.nl/~erwin/speechrecognition.html
   http://www.liacs.nl/~erwin/SR2003/ [See slides under Students/ and Workshops/]
   http://www.liacs.nl/~erwin/SR2005/
   http://www.liacs.nl/~erwin/SR2006/
   http://www.liacs.nl/~erwin/SR2009/ [also includes a workshop on HTK]
10. http://www.speech.cs.cmu.edu/comp.speech/ [infinite useful links]

11. Speech Recognition With CMU Sphinx [Blog by N. Shmyrev, one current Sphinx developer]

Download CMU SPHINX related files

1. Go to http://sourceforge.net/projects/cmusphinx/files/ There we find all the versions of all software related to Sphinx[1]

   We need Sphinx3, SphinxTrain, Sphinxbase and CMUCLMTK.

   Sphinx3 is the speech recognizer (decoder).

   SphinxTrain is a set of tools for acoustic modeling.

   SphinxBase is a common set of library used by several projects in CMU Sphinx.

   CMU-Cambridge Language Modeling Toolkit is a suite of tools which carry out language model training. *Source: 'Hieroglyphs'*

   Direct download links from sourceforge

   Sphinx3-0.8

   SphinxTrain-1.0

   Sphinxbase0.6.1

   CMUCLTK (just the binaries)

   (***Prefer these***) Local (IITB) links

   Sphinx3-(downloaded from SVN repos on 9th March 2011)

   SphinxTrain-1.0(downloaded from Sourceforge)

   Sphinxbase (downloaded from SVN repos on 9th March 2011)

   CMUCLMTK (binaries created from SVN version on 9th March 2011)

2. Create a folder called **sphinx** in C:\ drive (for that matter you can choose any drive). Save the 4 *.zip files under **C:\sphinx\**

Extract the files

1. Use Win-zip to extract the zip files. **Right click** on each of them and select **Win-zip** > **Extract to here**.
   This will create following folders-

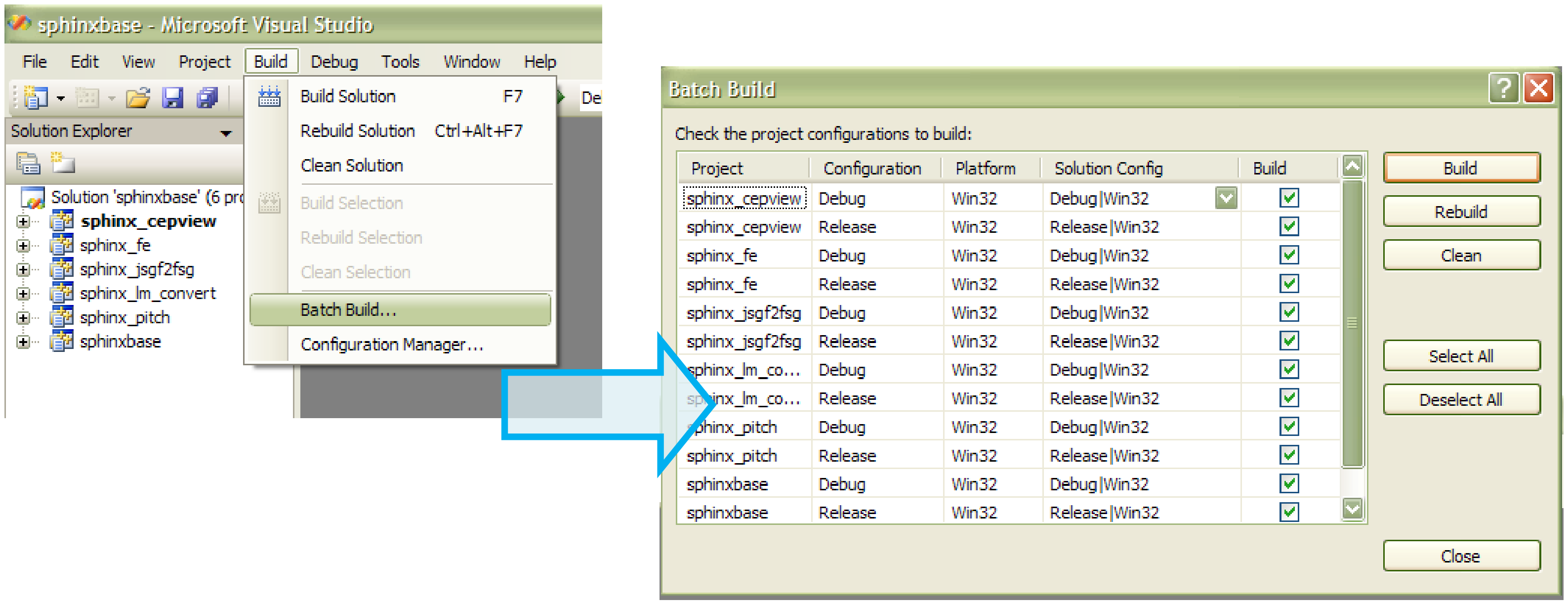
   C:\sphinx\sphinx3
   C:\sphinx\sphinxtrain
   C:\sphinx\sphinxbase
   C:\sphinx\cmuclmtk

[1] Latest (bleeding-edge) versions can be downloaded from this svn repository.

First of all, we will build **sphinxbase**, because next installations are dependent on it.

Install sphinxbase

1. Open **C:\sphinx\sphinxbase**
2. Doubleclick on **sphinxbase.sln**, the Visual Studio solution file for sphinxbase (You should have Visual Studio 2008 or newer)
3. In the menu, select **Build** -> **Batch Build.** Click **Select All** and then **Build** in the Batch Build window. Close the project after successful build.



Check the project configurations to build:

Project	Configuration	Platform	Solution Config	Build
sphinx_cepview	Debug	Win32	Debug|Win32	✓
sphinx_cepview	Release	Win32	Release|Win32	✓
sphinx_fe	Debug	Win32	Debug|Win32	✓
sphinx_fe	Release	Win32	Release|Win32	✓
sphinx_jsgf2fsg	Debug	Win32	Debug|Win32	✓
sphinx_jsgf2fsg	Release	Win32	Release|Win32	✓
sphinx_lm_co...	Debug	Win32	Debug|Win32	✓
sphinx_lm_co...	Release	Win32	Release|Win32	✓
sphinx_pitch	Debug	Win32	Debug|Win32	✓
sphinx_pitch	Release	Win32	Release|Win32	✓
sphinxbase	Debug	Win32	Debug|Win32	✓
sphinxbase	Release	Win32	Release|Win32	✓

Figure 1- Building Sphinxbase

4. This will create following 5 exe files, sphinxbase.dll and sphinxbase.lib files in **C:\sphinx\sphinxbase\bin\Debug (also in C:\sphinx\sphinxbase\bin\Release)**

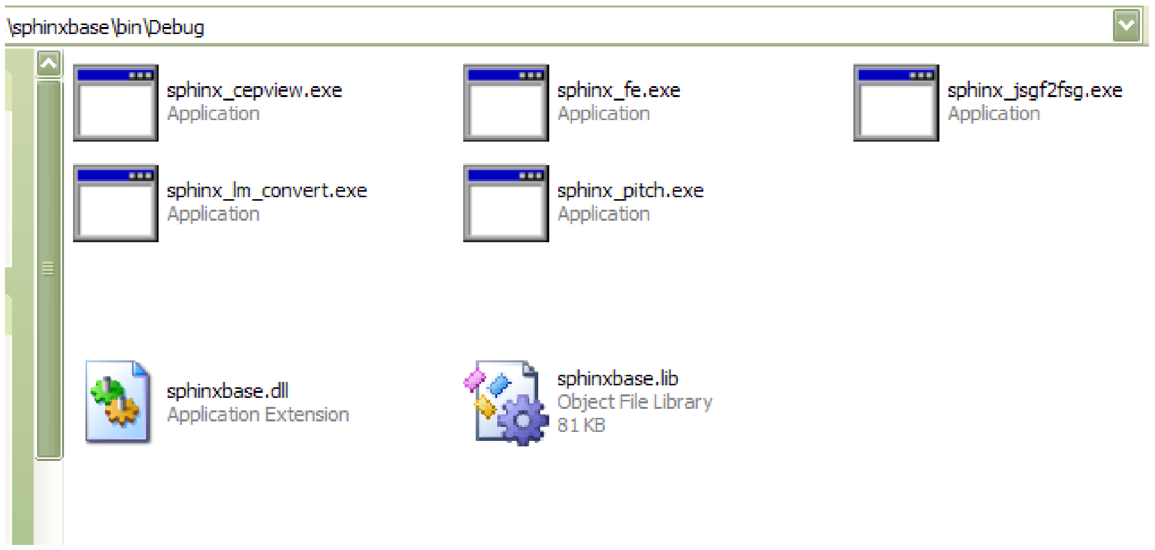


Figure 2 Executables and other files in sphinxbase

Difference between **debug** and **release** version of executables (we chose to create both above)

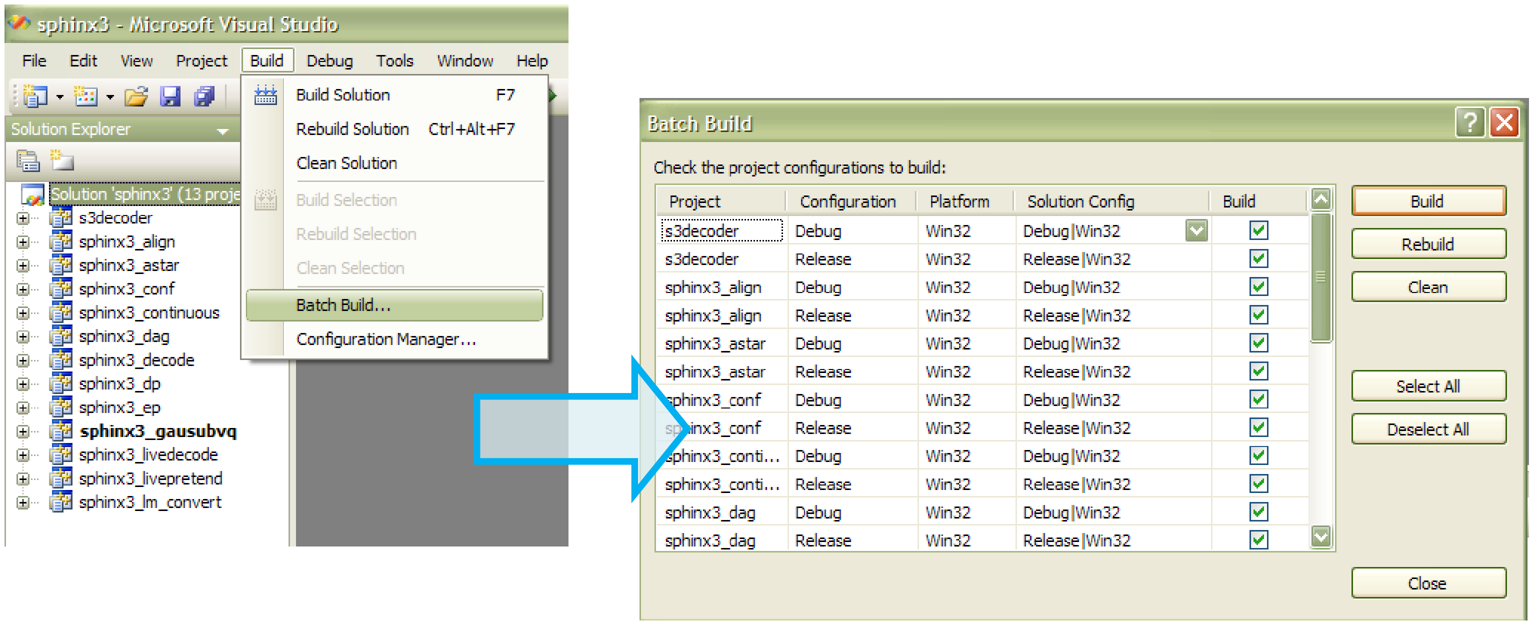
Debug and Release are different configurations for building your project.

You generally use the Debug mode for debugging your project, and the Release mode for the final build for end users. The Debug mode does not optimize the binary. It produces (as optimizations can greatly complicate debugging), and generates additional data to aid debugging. The Release mode enables optimizations and generates less (or no) extra debug data. Source: 1 and 2

See next page for Sphinx3 decoder installation.

Install sphinx3

1. Open **C:\sphinx\sphinx3**
2. Doubleclick on **sphinx3.sln.** Next build the projects same as for sphinxbase. Close the project after successful build.

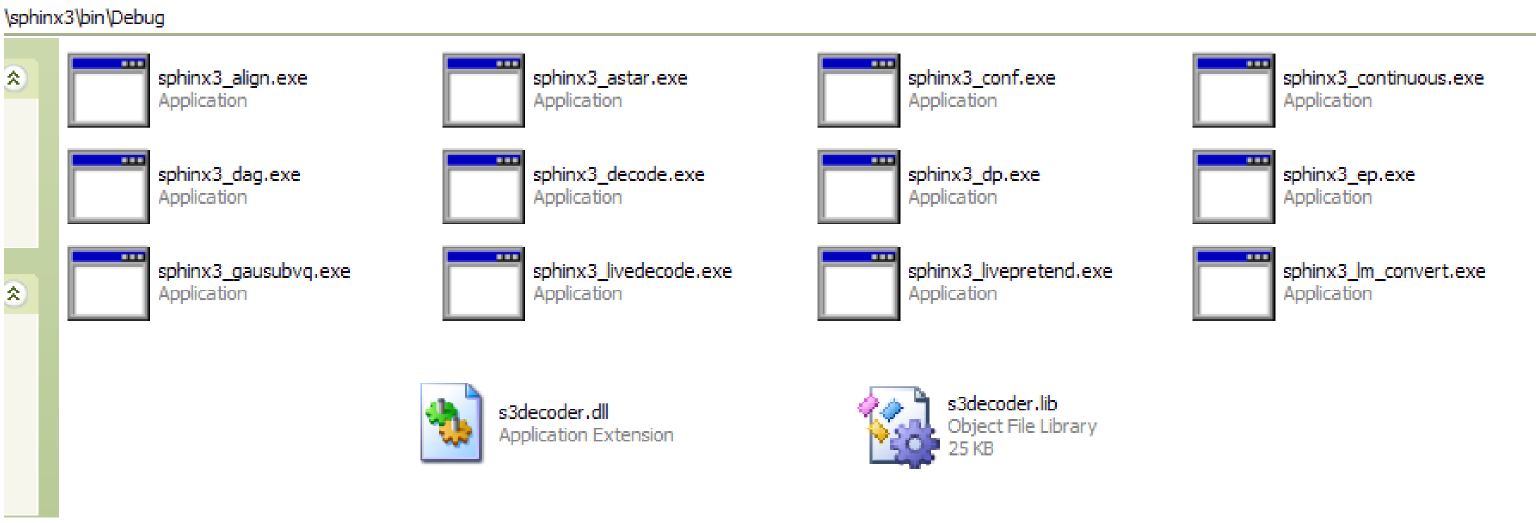


Check the project configurations to build:

Project	Configuration	Platform	Solution Config	Build
s3decoder	Debug	Win32	Debug|Win32	✓
s3decoder	Release	Win32	Release|Win32	✓
sphinx3_align	Debug	Win32	Debug|Win32	✓
sphinx3_align	Release	Win32	Release|Win32	✓
sphinx3_astar	Debug	Win32	Debug|Win32	✓
sphinx3_astar	Release	Win32	Release|Win32	✓
sphinx3_conf	Debug	Win32	Debug|Win32	✓
sphinx3_conf	Release	Win32	Release|Win32	✓
sphinx3_conti...	Debug	Win32	Debug|Win32	✓
sphinx3_conti...	Release	Win32	Release|Win32	✓
sphinx3_dag	Debug	Win32	Debug|Win32	✓
sphinx3_dag	Release	Win32	Release|Win32	✓

Figure 3 - Building Sphinx3

3. *Step 2* will create following 12 exe files, s3decoder.dll, s3decoder.lib and other files in **C:\sphinx\sphinx3\bin\Debug** and **C:\sphinx\sphinx3\bin\Release**



\sphinx3\bin\Debug

sphinx3_align.exe Application
sphinx3_astar.exe Application
sphinx3_conf.exe Application
sphinx3_continuous.exe Application
sphinx3_dag.exe Application
sphinx3_decode.exe Application
sphinx3_dp.exe Application
sphinx3_ep.exe Application
sphinx3_gausubvq.exe Application
sphinx3_livedecode.exe Application
sphinx3_livepretend.exe Application
sphinx3_lm_convert.exe Application
s3decoder.dll Application Extension
s3decoder.lib Object File Library 25 KB

Figure 4 - Executables and other files in Sphinx3

4. Copy C:\sphinx\sphinxbase\bin\Release\sphinxbase.dll to C:\sphinx\sphinx3\bin, C:\sphinx\sphinx3\bin\Debug and C:\sphinx\sphinx3\bin\Release

Install CMUCLTK

1. Open **C:\sphinx\cmuclmtk**
2. I have already provided compiled binaries (32bit) inside **C:\sphinx\cmuclmtk\executables**. Let me know if they give any error later.

Install SphinxTrain

This is not required for decoding. So we will defer its installation.

We are now done with building executables! How are we going to use them?

Before using them, note that it will be tiresome to copy and paste the *.exes to the location at which you will have the test data. So, in order to be able to call them from *anywhere* we will do following –

Set Path Variables

Windows-XP Users

1. Click **START,** move pointer over **My Computer,** right-click**,** select **properties.** This will open a **System Properties** window. [click to see Figure 5]
2. Select **Advanced** tab in the **System Properties** window. [Figure 6]
3. Click on **Environment Variables** button which is near the bottom of window. This will open *Environmental Variables* window.
4. Scroll down and select **PATH** in the **System Variables** box. Click **Edit**. [Figure 7]
5. Above step will throw up an **Edit System Variable** window. Name of the variable is **PATH.** [Fig 8]
6. Click near end of text in **Variable Value** box. We will add some paths here. Paths are separated by *semicolons* and no spaces occur anywhere.
7. Type a semicolon near end of last path in the box and write **C:\sphinx\sphinx3\bin\Release** after that. [Figure 9]
8. Exactly after that (without leaving any space) type another semicolon and write **C:\sphinx\sphinxbase\bin\Release**
9. Give another semicolon and write **C:\sphinx\cmuclmtk\executables**
   In short you have to add

;C:\sphinx\sphinx3\bin\Release;C:\sphinx\sphinxbase\bin\Release;C:\sphinx\cmuclmtk\executables

10. Click **OK.** *Edit System Variable* window will disappear**.** Click another **OK.** *Environmental Variable* window will disappear. Click one more **OK** and let *System Properties* window disappear.
11. Restart the computer (may not be needed, still.)

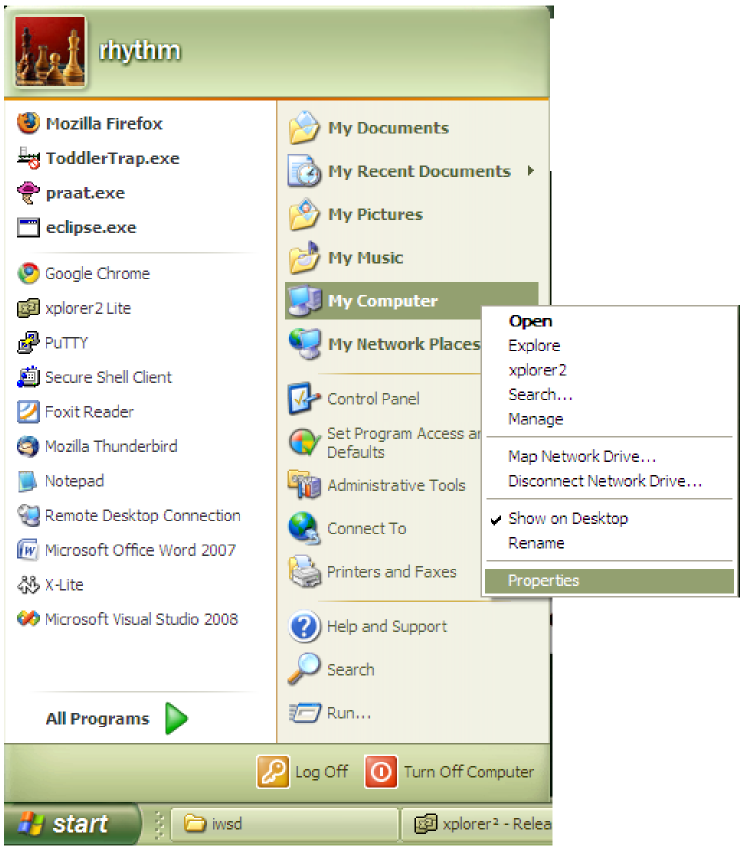
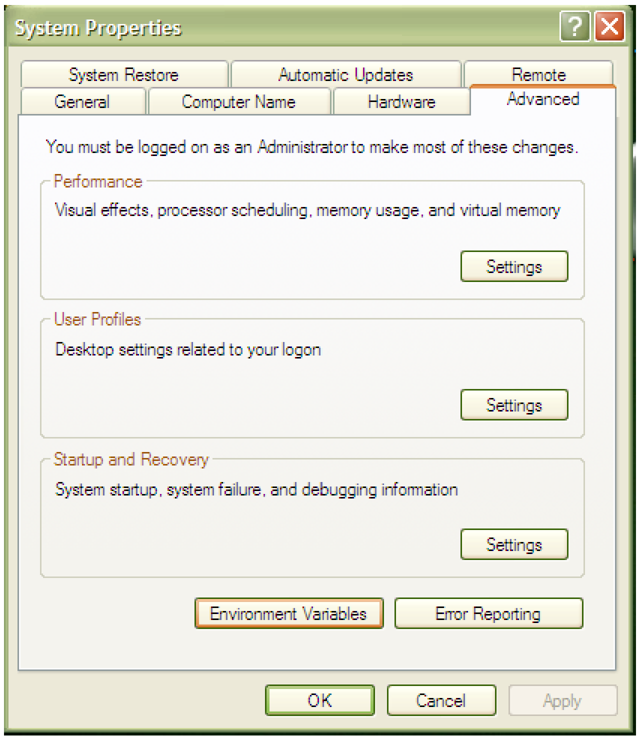

rhythm
Mozilla Firefox
ToddlerTrap.exe
praat.exe
eclipse.exe
Google Chrome
xplorer2 Lite
PuTTY
Secure Shell Client
Foxit Reader
Mozilla Thunderbird
Notepad
Remote Desktop Connection
Microsoft Office Word 2007
X-Lite
Microsoft Visual Studio 2008
All Programs
My Documents
My Recent Documents
My Pictures
My Music
My Computer
My Network Places
Control Panel
Set Program Access and Defaults
Administrative Tools
Connect To
Printers and Faxes
Help and Support
Search
Run...
Open
Explore
xplorer2
Search...
Manage
Map Network Drive...
Disconnect Network Drive...
Show on Desktop
Rename
Properties
Log Off
Turn Off Computer
start
iwsd
xplorer² - Relea

Figure 5



You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

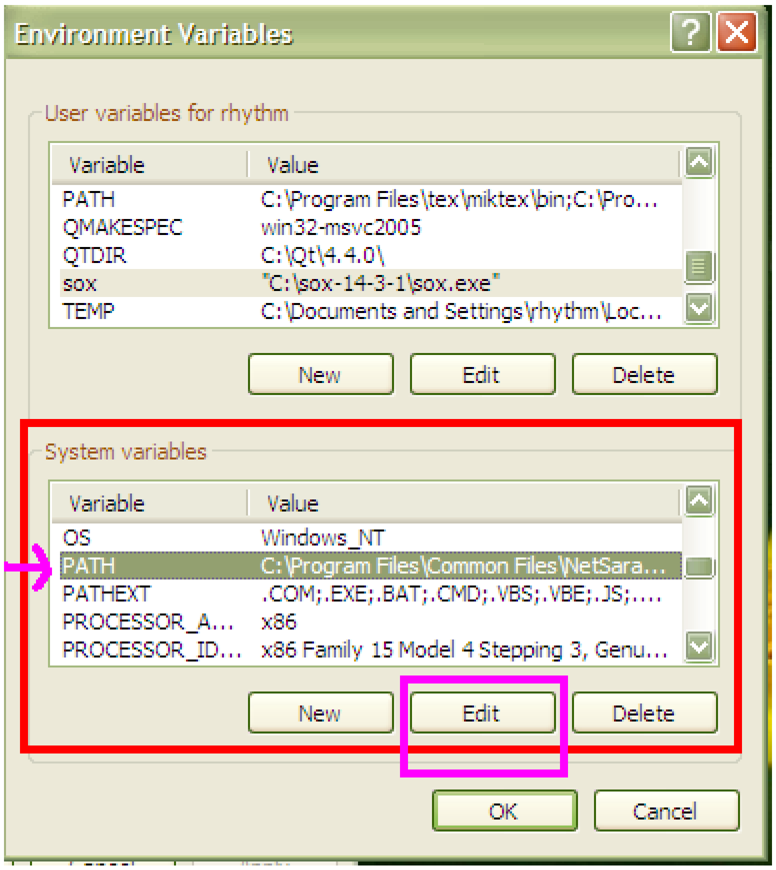
User Profiles

Desktop settings related to your logon

Startup and Recovery

System startup, system failure, and debugging information

Figure 6



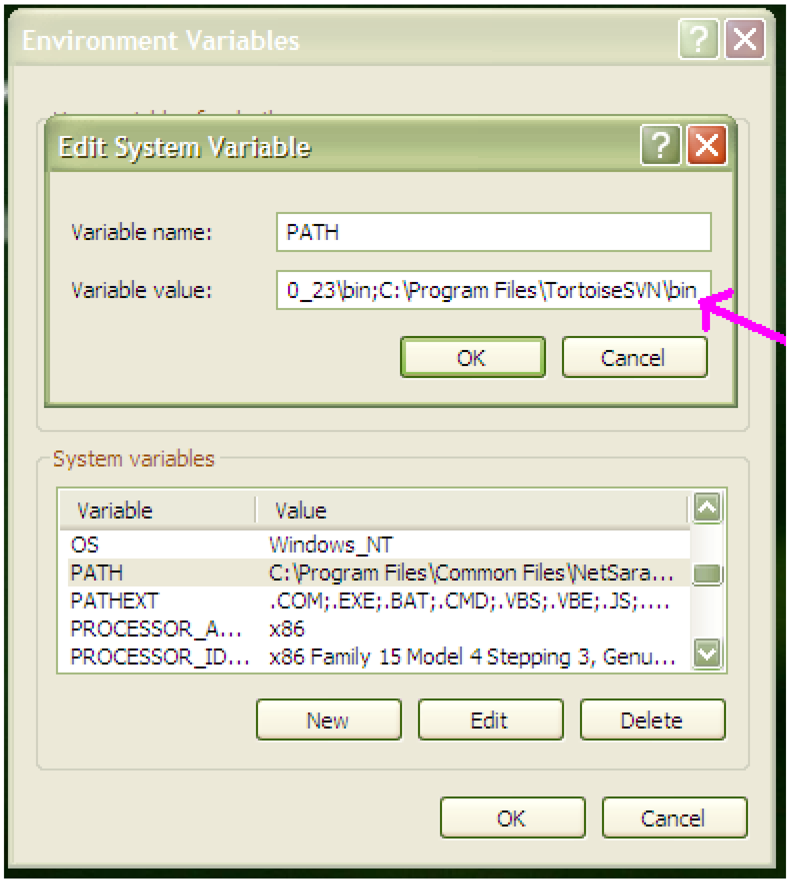
User variables for rhythm

Variable	Value
PATH	C:\Program Files\tex\miktex\bin;C:\Pro...
QMAKESPEC	win32-msvc2005
QTDIR	C:\Qt\4.4.0\
sox	"C:\sox-14-3-1\sox.exe"
TEMP	C:\Documents and Settings\rhythm\Loc...

System variables

Variable	Value
OS	Windows_NT
PATH	C:\Program Files\Common Files\NetSara...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 15 Model 4 Stepping 3, Genu...

Figure 7


Edit System Variable

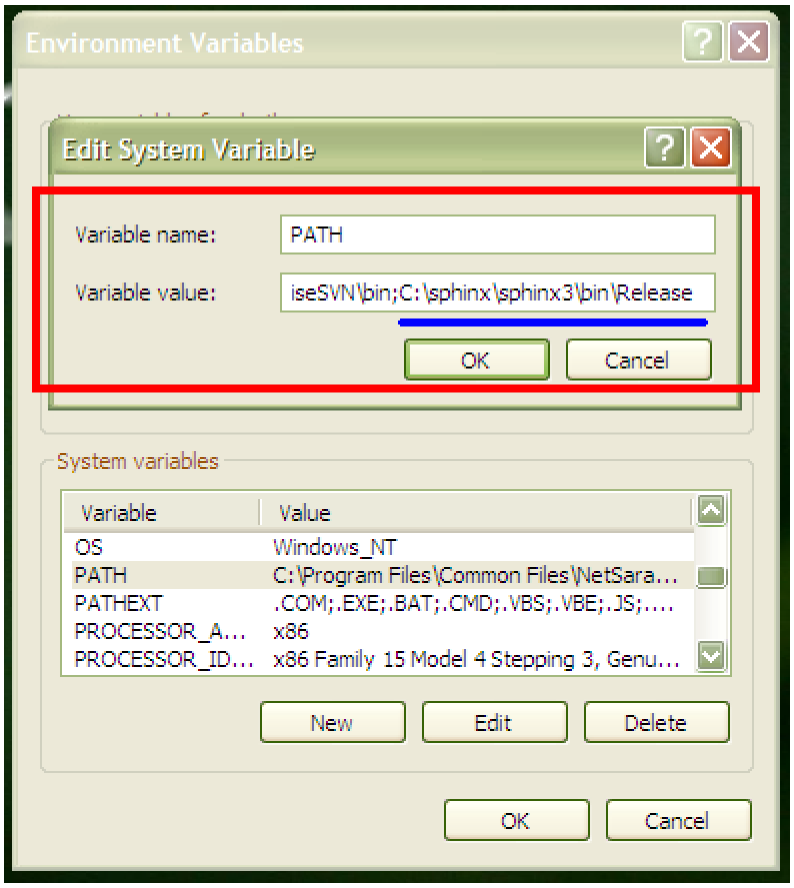
Variable name: PATH

Variable value: 0_23\bin;C:\Program Files\TortoiseSVN\bin

System variables

Variable	Value
OS	Windows_NT
PATH	C:\Program Files\Common Files\NetSara...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 15 Model 4 Stepping 3, Genu...

Figure 8



Variable name: PATH

Variable value: iseSVN\bin;C:\sphinx\sphinx3\bin\Release

Variable	Value
OS	Windows_NT
PATH	C:\Program Files\Common Files\NetSara...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...	x86
PROCESSOR_ID...	x86 Family 15 Model 4 Stepping 3, Genu...

Figure 9

Windows 7 Users

1. Click **Windows button,** move pointer over **Computer,** right-click, select **Properties.** [Figure 10]
2. Above step will throw up a window. Click on **Advanced system settings** in the left column. This will show the **System Properties** window. [Figure 11]
3. Click on **Advanced** tab. Click **Environment Variables** button. [Figure 12]
4. Scroll down and select **Path** in the **System Variables** box.
5. Click **Edit.** [Figure 13]
6. **After this follow the same steps as for Windows XP users [Steps 6 - 11]**

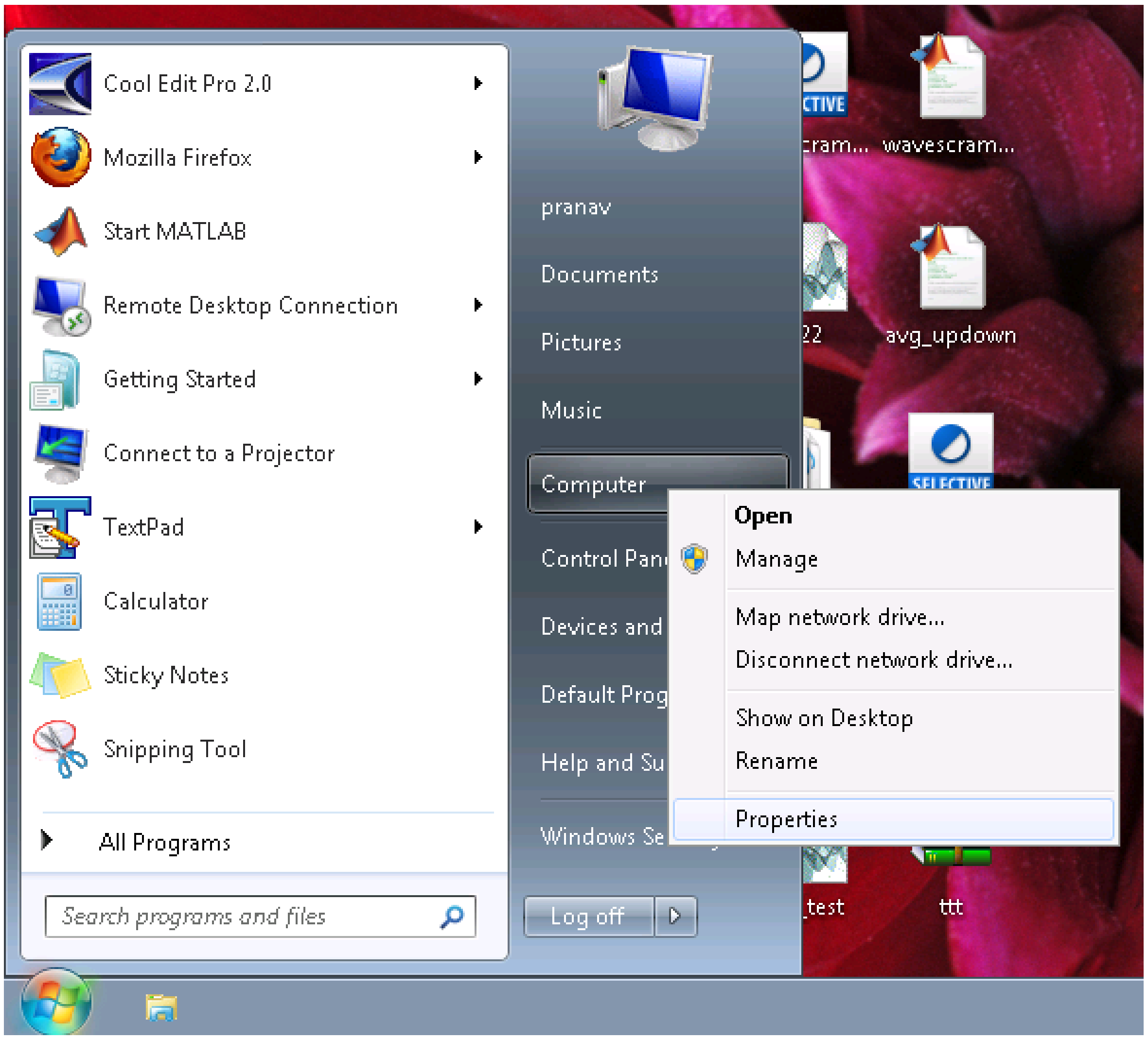

Cool Edit Pro 2.0
Mozilla Firefox
Start MATLAB
Remote Desktop Connection
Getting Started
Connect to a Projector
TextPad
Calculator
Sticky Notes
Snipping Tool
All Programs
Search programs and files
pranav
Documents
Pictures
Music
Computer
Control Pan
Devices and
Default Prog
Help and Su
Windows Se
Log off
Open
Manage
Map network drive...
Disconnect network drive...
Show on Desktop
Rename
Properties
wavescram...
avg_updown
test
ttt

Figure 10

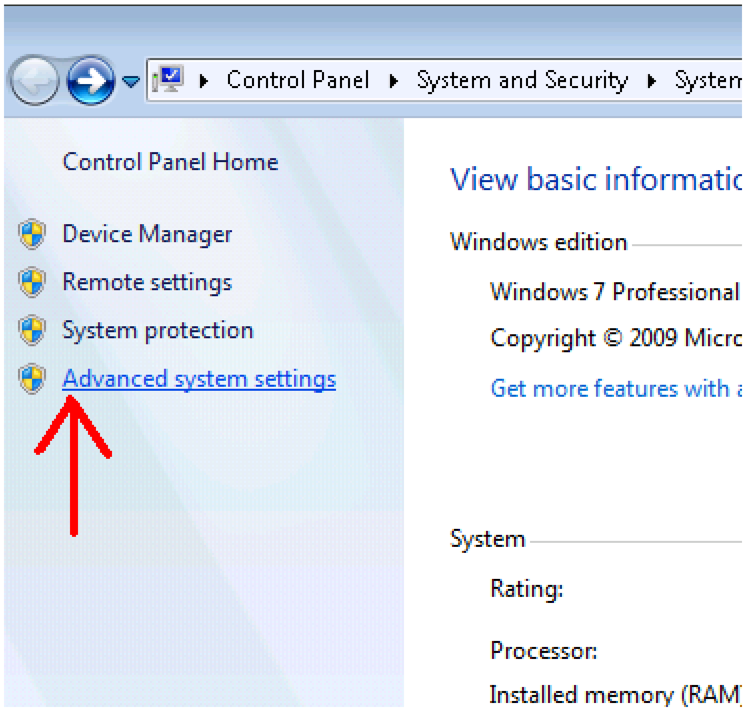
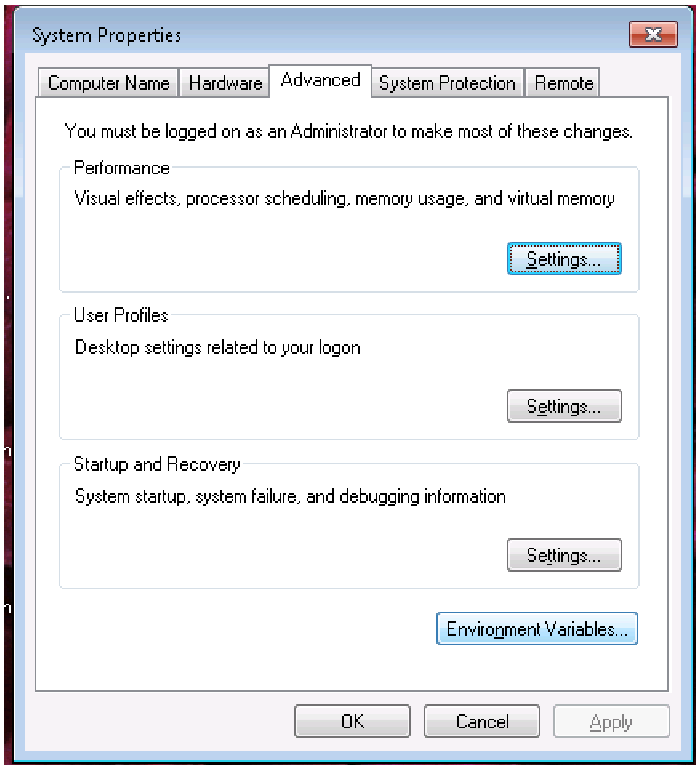


Figure 11



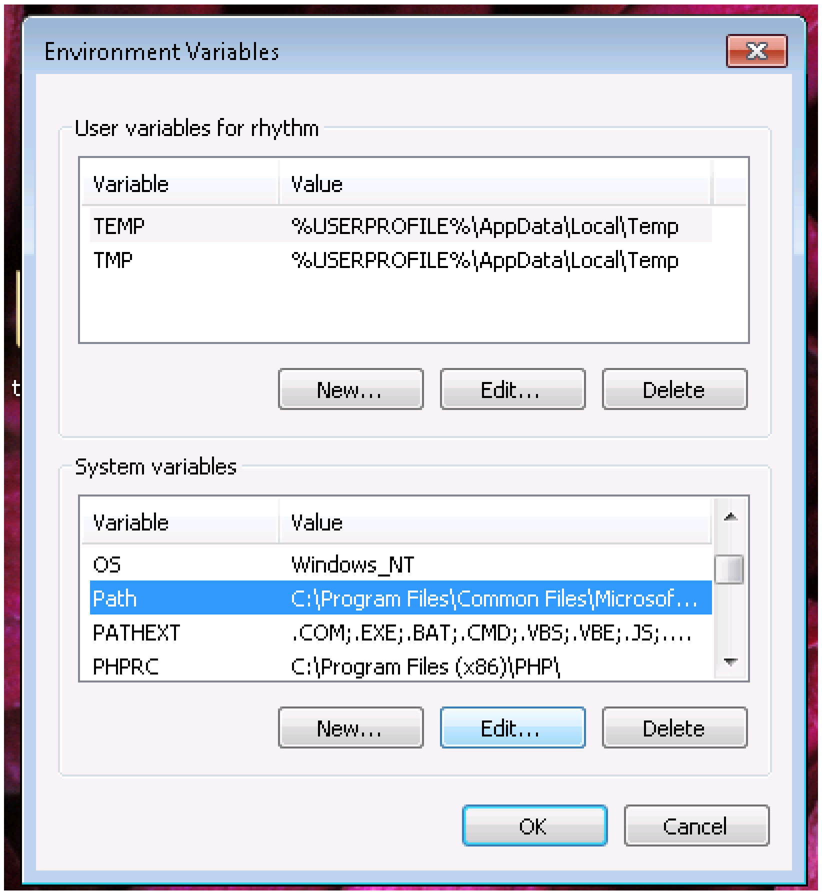
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

Figure 12



Variable	Value
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Variable	Value
OS	Windows_NT
Path	C:\Program Files\Common Files\Microsof...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PHPRC	C:\Program Files (x86)\PHP\

Figure 13

>> IMPORTANT <<

In order to check that all the paths have been set correctly open the command prompt. Type **sphinx3_decode** and hit Enter. If it gives following message

“'sphinx3_decode' is not recognized as an internal or external command, operable program or batch file.”

Then there was some mistake in setting path variables for **sphinx3.**

Similarly test using following two commands to see whether paths of **sphinxbase** and **cmuclmtk** are set correctly.

sphinx_fe [an executable in sphinxbase]

and

text2idngram [an executable in cmuclmtk]

Things we need for decoding

First create a workspace (a directory wherein test data etc. will be located).
Create a new folder called **sphinxtest** anywhere on computer. I created it here

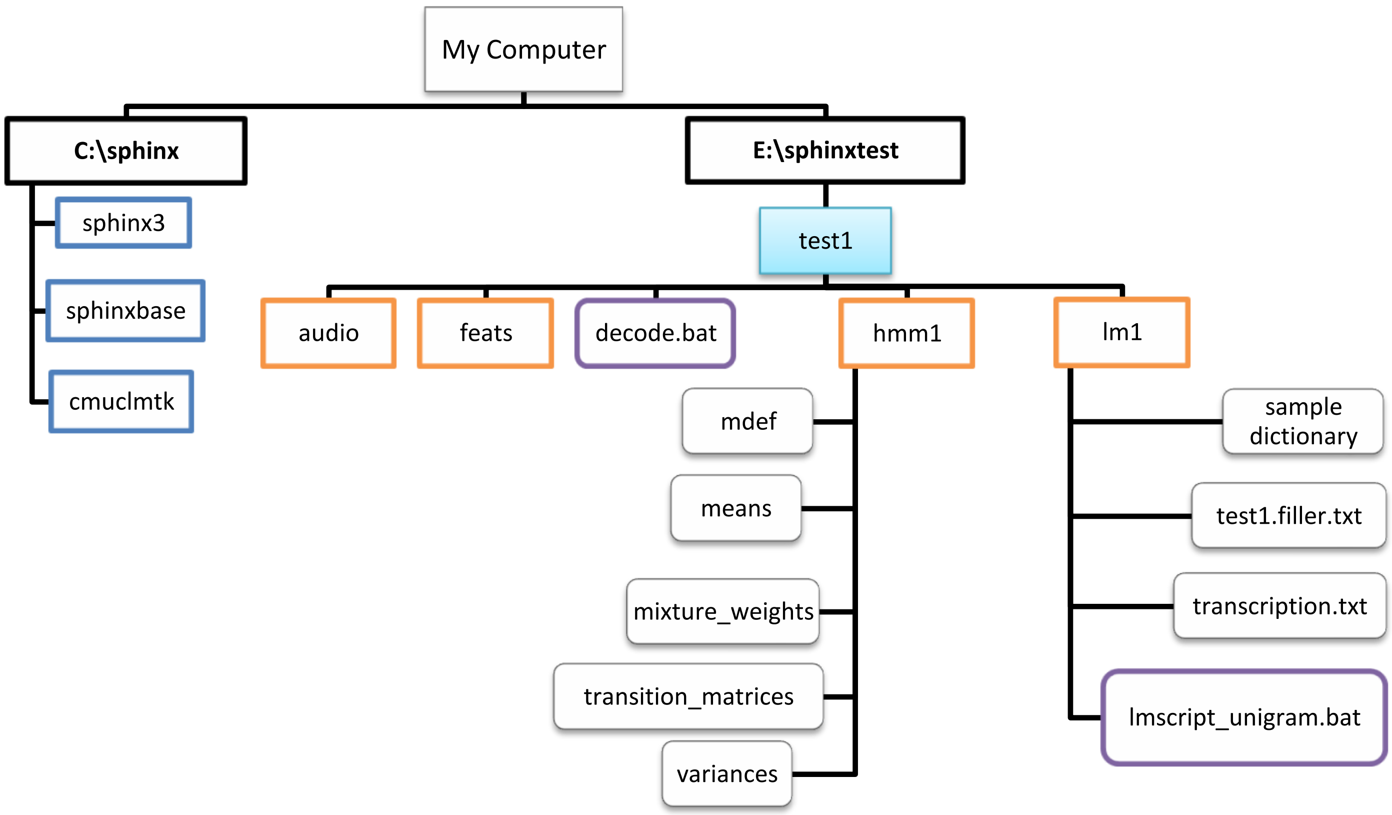
E:\sphinxtest

We need following things for decoding -

1. **Audio files**
   - Download a zipped test folder “test1.zip” them from here and keep it in **E:\sphinxtest**
   2. Right click on test.zip and select **Win-zip** > **Extract to here**.
   - Go to **E:\sphinxtest\test1\audio**. Here you will see the wav files which we will use for testing.
2. **Acoustic Models**
   - These are present in **E:\sphinxtest\test1\hmm1** folder.
3. **Dictionary**
4. **Language model**

   We will create **3** and **4** as explained later.

Till now we have following directory structure



Let’s create the Dictionary!

Listen to the **cerii.wav** in **E:\sphinxtest\test1\audio**. What does it say?

It contains the word – **Cherry**. Similarly other wav files have been named according to what they contain.

We need to tell the decoder *which* words it is supposed to recognize and what is the phone sequence corresponding to each of those words. This is accomplished by the pronunciation dictionary. Here our dictionary will contain these 5 words- ananasa, baajari, bhaat, cherry, makaa.

Create an empty file and write following lines in it

```
ananasa        a n a n a s
baajarii       b aa j r ii
baajarii(2)    b a j a r ii
bhaat          bh aa t
cherry         c e r ii
makaa          m a k aa
```

The first column is the **word** and second column is the corresponding **phonetic representation**. In each line, after writing the word, give TAB and write phones one after another with SINGLE SPACE between them e.g. **cherry**TAB**c**SPACE**e**SPACE**r**SPACE**ii**

Save this file as **test1.dic.txt** in **E:\sphinxtest\test1\lm1** (.txt extension is NOT necessary). [I have already provided this file- Pranav]

From where do these phones come?
It depends on which phone models were created during training. If you open the model definition file- **E:\sphinxtest\test1\hmm1\mdef** with your text editor (I recommend TextPad for clear formatting, please *avoid* notepad), you will see list of all the phones, fillers and corresponding HMM state ids.

What do these phones sound like?

Open the file **labelSetASR100815.pdf** in the **test1** folder. It lists all the phones in **mdef** file + some extra phones (e.g. l’) and example words for each of them. You can add your own entries in the dictionary if you wish.

Note: Some words may have more than one possible pronunciation. To recognize all the common pronunciation variants we list them all in the dictionary. Here **baajarii(2)** is the second possible pronunciation of **baajarii.** If you come up with a third one, you can write it as **baajarii(3)** and so on.

Also note that all the words/phones in the dictionary are lowercase. Instead they can all be uppercase, just make sure that you don’t mix them. CHERRY and cherry are the same!

What is the Filler dictionary?

According to Wikipedia *a **filler** is a sound or word that is spoken in conversation by one participant to signal to others that he/she has paused to think but is not yet finished speaking.* Examples include umm, eh etc. In general, filler is anything which is used to fill the gaps. In speech recognition we create models for fillers which (if deemed right) are inserted by the decoder in its hypothesis about the audio input.

Open the file **test1.filler.txt** (in **E:\sphinxtest\test1\lm1**) with TextPad (NOT notepad). You will see these entries-

```
+AIR+         +AIR+
+BABBLE+      +BABBLE+
+CAR_HORN+    +CAR_HORN+
+THROAT+      +THROAT+
+BG_NOISE+    +BG_NOISE+
```

It is obvious what these fillers sound like (AIR -> sound of flowing air etc.).

Apart from filler sounds, these three lines are there-

```
<s>     SIL
</s>    SIL
<sil>   SIL
```

These lines are common to *any* filler dictionary. <s> denotes start of the sentence (utterance) silence, </s> denotes the end silence. In the decoder log file, you will see just <sil>. All 3 corresponds to **SIL** i.e. silence.

Now for the Language Model...

Quoting http://cmusphinx.sourceforge.net/wiki/tutoriallm (a tutorial on building language models)

"There are two types of models that describe language - grammars and statistical language models. Grammars describe very simple types of languages for command and control, and they are usually written by hand or generated automatically with plain code."

Here we will create a statistical language model (called as **n-gram**) using **CMUCLMTK** (the CMU-Cambridge Statistical Language Modeling Toolkit).

A little theory –

Let **W** be a sequence of words ($w_1, w_2, \ldots, w_m$) in the dictionary. P(W) is the probability of occurrence of this sequence. We can write P(W) as –

$$P(W) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)\ldots P(w_m \mid w_1, w_2, \ldots w_{m-1}) = \sum_{i=1}^{m} P(w_i \mid \Phi_i)$$

Φ_i is in some sense the *history* of the ith word.

Note we could also have expanded P(W) as below but (perhaps) it makes no difference.

$$P(W) = P(w_m)P(w_{m-1} \mid w_m)P(w_{m-2} \mid w_m, w_{m-1})\ldots P(w_1 \mid w_m, w_{m-1}, \ldots w_2) = \sum_{i=1}^{m} P(w_i \mid \Phi'_i)$$

In n-gram model we assume that the history of a word is only composed of last **n-1** words. An n-gram model specifies probability of occurrence of n-grams (group of n consecutive words).

Unigram language model
Here n = 1 and we expand P(W) as

$$P(W) = P(w_1)P(w_2)P(w_3)\ldots P(w_m) = \sum_{i=1}^{m} P(w_i)$$

So the occurrence of each word is independent of any other word. Further assume that each of $P(w_i)$ is equal (all words equiprobable).

Bigram / trigram language models

In bigram Φ_i is composed of 1 previous word, and in trigram it is composed of 2 previous words.

Example from here - The LM probability of an entire sentence is the product of the individual word probabilities. For example, the LM probability of the sentence "HOW ARE YOU" is:

P(HOW | <s>)*P(ARE | <s>, HOW)*P(YOU | HOW, ARE)*P(</s> | ARE, YOU)

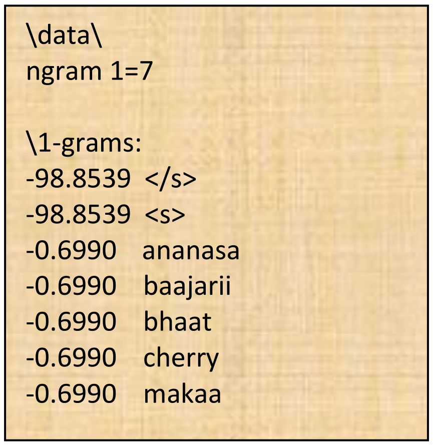
Let's now look at an example of language model to see what it means.

There is a batch script lmscript_unigram.bat in lm1 folder. To run it, open the command prompt and go to **E:\sphinxtest\test1\lm1**

Now run **lmscript_unigram.bat** in command prompt.

It will give a message "7 unigrams created" and will create unigram model **test1.lm**.

Open **test1.lm** with a text-editor.



```
\data\
ngram 1=7

\1-grams:
-98.8539  </s>
-98.8539  <s>
-0.6990   ananasa
-0.6990   baajarii
-0.6990   bhaat
-0.6990   cherry
-0.6990   makaa
```

test1.lm

First column gives log10 probability of observing the word written next. [$10^{-0.699}$ = ~0.2]
Note that language model doesn't contain any of the fillers (but <s> and </s> are mandatory).

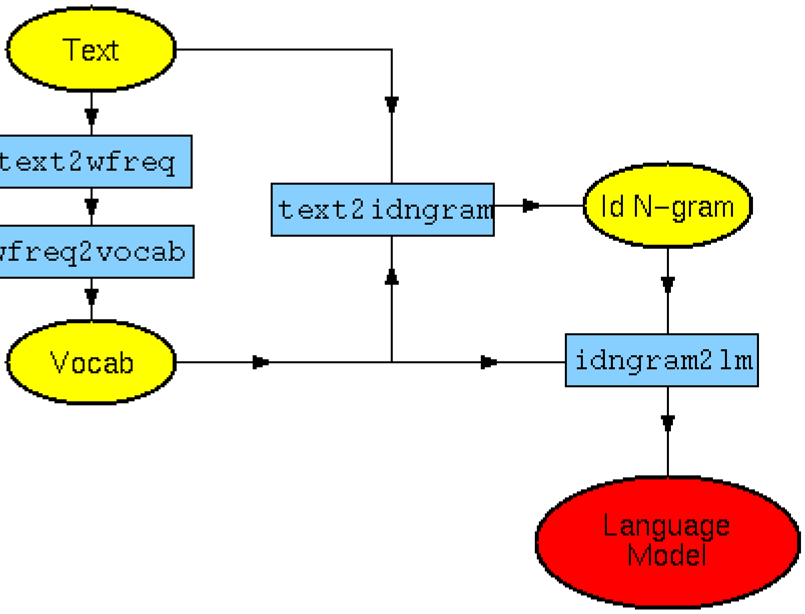


Figure 14 - Language model creation (from http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html#typical_use)

Figure 14 shows the steps for creating a language model.

"Text" corresponds to **transcription.txt** which contains the transcription using which unigram model is being trained.

"Vocab" corresponds to **test1.vocab** (open it with TextPad). It contains alphabetical list of all the words (excluding fillers, but including context cues [**test1.css.txt**])

"**Id N-gram**" corresponds to **test1.idngram**.

test1.lm is the language model file. It has to be converted into binary DMP format (**test1.lm.DMP**) for it to be readable by sphinx3 decoder.
Try to see which steps in the **Figure 11** correspond to which commands in **lmscript_unigram.bat**.

It's time to decode.

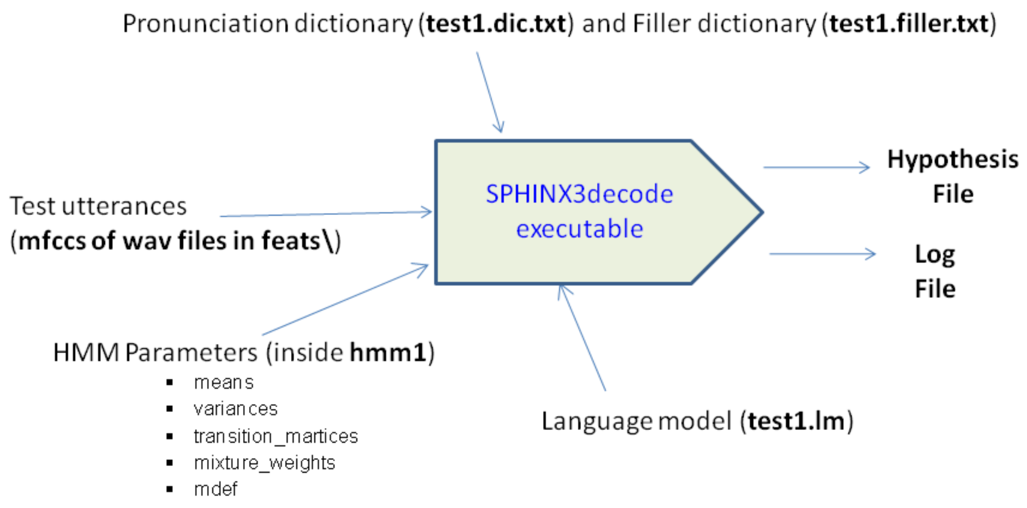

Pronunciation dictionary (test1.dic.txt) and Filler dictionary (test1.filler.txt)
Test utterances (mfccs of wav files in feats\)
SPHINX3decode executable
Hypothesis File
Log File
HMM Parameters (inside hmm1)
means
variances
transition_martices
mixture_weights
mdef
Language model (test1.lm)

Figure 15 "Inputs and outputs" Sphinx3_decode

As shown in **Figure 15** we need MFCCs of the wav files which we wish to recognize. Open the script **decode.bat** in **test1** folder using a text editor. It has two main commands –

1. Using **sphinx_fe** MFCCs are computed for all the files whose names appear in **E:\sphinxtest\test1\list**
   All the wav files are in **E:\sphinxtest\test1\audio**
   Following parameters are provided to **sphinx_fe**

Parameter	Value set in decode.bat and its meaning
alpha	**0.97** (pre-emphasis factor)
samprate	**8000** (Hz)
dither	**Yes** (add ½ bit noise)
doublebw	**No** (Do not use double bandwidth filters)
nfilt	**36** (number of filters used in MFCC computation)
ncep	**13** (number of cepstral coefficients)
lowerf	**133.33** (Lower cutoff frequency in Hz)
upperf	**3500** (Upper cutoff frequency in Hz)
nfft	**256** (256 point FFT)
wlen	**0.0256** (Hamming window length in seconds)
frate	**100** (frames per second)
c	**E:\sphinxtest\test1\list** (control file, contains names of wav files w/o .wav extension)
di	**E:\sphinxtest\test1\audio** (wav files are assumed to be present here)
ei	**wav** (extension of input audio files)
mswav	**Yes** (whether input files are in mswav format)
do	**E:\sphinxtest\test1\feats** (directory where MFCC files will be stored)
eo	**mfc** (extension of MFCC files)
mfcclog	**E:\sphinxtest\test1\mfcclog.txt** (log file created by **sphinx_fe**)

2. Secondly, we use **sphinx3_decode**, we recognize all the wav files specified in the control file ("**list**"). We have specified following parameters for the decoder [there are many more params that we have not specified. To see them just type **sphinx3_decode** in command window and hit Enter]–

Parameter	Value set in decode.bat and their meaning
hmm	**E:\sphinxtest\test1\hmm1** (folder where the 5 parameter files of acoustic models are present)
lm	**E:\sphinxtest\test1\lm1\test1.lm.DMP** (path to the binary language model file)
dict	**E:\sphinxtest\test1\lm1\test1.dic.txt** (path to pronunciation dictionary)
fdict	**E:\sphinxtest\test1\lm1\test1.filler.txt** (path to filler dictionary)
hyp	**E:\sphinxtest\test1\decode.out.txt**(decoder hypothesis will be written here)
cepdir	**E:\sphinxtest\test1\feats** (folder where the MFCC files of test data are present)
cepext	**.mfc** (extension of MFCC files)
ceplen	**13** (number of cepstral coefficients used in creating MFCC files)
frate	**100** (frame rate, in frames per second used while creating MFCC files)
ctl	**E:\sphinxtest\test1\list** (control file, list of files to decode)
dither	**yes**
hypseg	**E:\sphinxtest\test1\hypseg (**Recognition result file, with word segmentations and scores. for more refer this)
outlatdir	**E:\sphinxtest\test1\lat** (folder in which to dump lattices. Lattice is a word-graph of all possible candidate words recognized during the decoding of an utterance, including other attributes such as their time segmentation and acoustic likelihood scores. for more refer this)
outlatfmt	**s3** (format in which to dump word lattices, either 's3' or 'htk')
latext	**lat** (filename extension for lattice files for more refer this and this)
hmmdump	**No** (If set to yes, we can see info about active HMM states for *each frame*)
logfn	**E:\sphinxtest\test1\decodelog.txt** (log file of decoding)

Note!!! Edit the first line in **decode.bat** if path to **test1** is different from **E:\sphinxtest\test1** on your computer.

Let's now run **decode.bat**. Open command window, CD to **E:\sphinxtest\test1\**
Type **decode.bat** and hit Enter.

Looking at the hypothesis file
Open **E:\sphinxtest\test1\decode.out.txt.** In each line the *word in bracket* is the name of the wav file and the *words before it* is the decoder output. For example, if a line reads **cherry cherry (cerii(-21db)_cerii),** it means **cerii(-21db)_cerii.wav** was recognized as **cherry cherry**.

Listen to each of the listed audio files and see which once were correctly recognized, partially correctly recognized, totally incorrectly recognized or not recognized at all.

Note that the hypothesis file doesn't contain information about inserted fillers, to see them we have to look into the log file.

Looking at the log file

Open **E:\sphinxtest\test1\decodelog.txt**

Initial part of log file gives info about default decoder parameters and their changed values (if any). Then there is a lot of information about how the decoder interprets acoustic models and language model.

Then you will see a **Backtrace information** about each of the wav files. Here is how to interpret a sample backtrace.

```
Backtrace(cerii(-21db)_cerii)
FV:cerii(-21db)_cerii>   WORD         SFrm  EFrm  AScr(UnNorm)  LMScore  AScr+LScr   AScale
fv:cerii(-21db)_cerii>   <sil>         0     12     277164       -74111    203053    394797
fv:cerii(-21db)_cerii>   cherry        13    54    -139331       -53781   -193112    446625
fv:cerii(-21db)_cerii>   +CAR_HORN+    55    76     240803       -74111    166692    607758
fv:cerii(-21db)_cerii>   cherry        77    118    23905        -53781   -29876     461520
fv:cerii(-21db)_cerii>   +BABBLE+      119   130    273888       -74111    199777    408987
FV:cerii(-21db)_cerii>   TOTAL                      676429      -329895
```

The **cerii(-21db)_cerii.wav** file has 131 frames (at framerate = 100 fps).

SFrm = strat frame index

EFrm = end frame index

AScr = acoustic score for the segment P (O|W)

LMScore = language model score

<sil> i.e. silence was recognized from frame[0] to frame[12]

cherry was recognized from frame[13] to frame[54]

and so on ..

In particular, listen to **cerii(-40dB)_cerii.wav** and **cerii(-50dB).wav** and see the decoder output. Can you hear 2 "cherries" in the first file and one cherry in second? Decoder can even recognize words which are of very low amplitude.

Also listen to **cerii_horn.wav** and see its backtrace in the log file. Car horn would have been recognized, see if its location has been correctly recognized.

Nothing has been recognized (apart from fillers) for **ananas_infy_zero.wav** even though you can make out what is being said.

Record your own wav files, put them in **audio** folder, add their name in **list** file and run **decode.bat** again.

-------------- Maths behind scores, I will update this section later ------------------------------------
If **O** is the observation vector
W' is the recognized word for given **O**

W' = argmax P(O|W)*P(W)

log [P(O|W)*P(W)] = **log [P(O|W)] + log [P(W)]**

log [P(W)] comes from the language model. It is equal to **Language_Weight*LMScore**
Default value of language_weight is 9.5.
Also sphinx uses log to the base 1.0001 while giving scores.

If **Q** is the phone sequence

P(O|W) = P(O|Q)*P(Q|W)

taking log
log [P(O|W)] = log [P(O|Q)*P(Q|W)] = **log [P(O|Q)] + log [P(Q|W)]**

P [O|Q] = Probability of observing **O** if **Q** were the phone sequence
---