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.

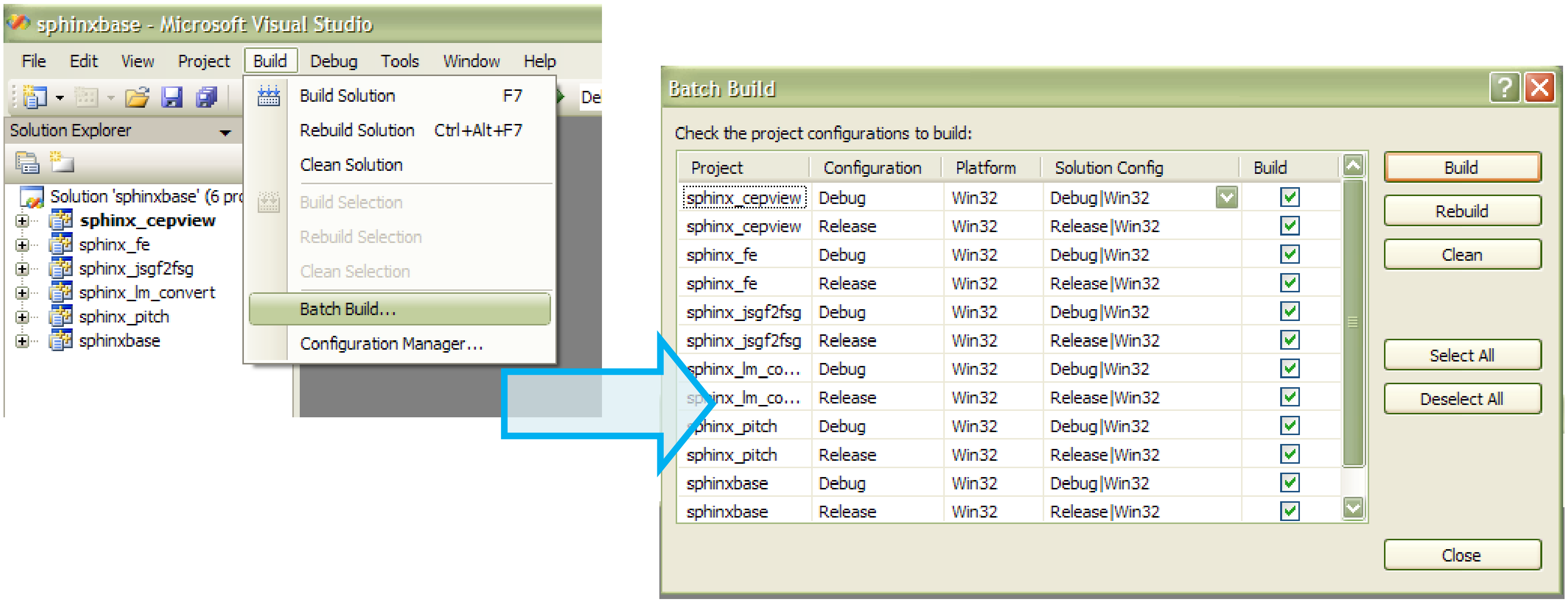


Figure 1- Building Sphinxbase

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)**

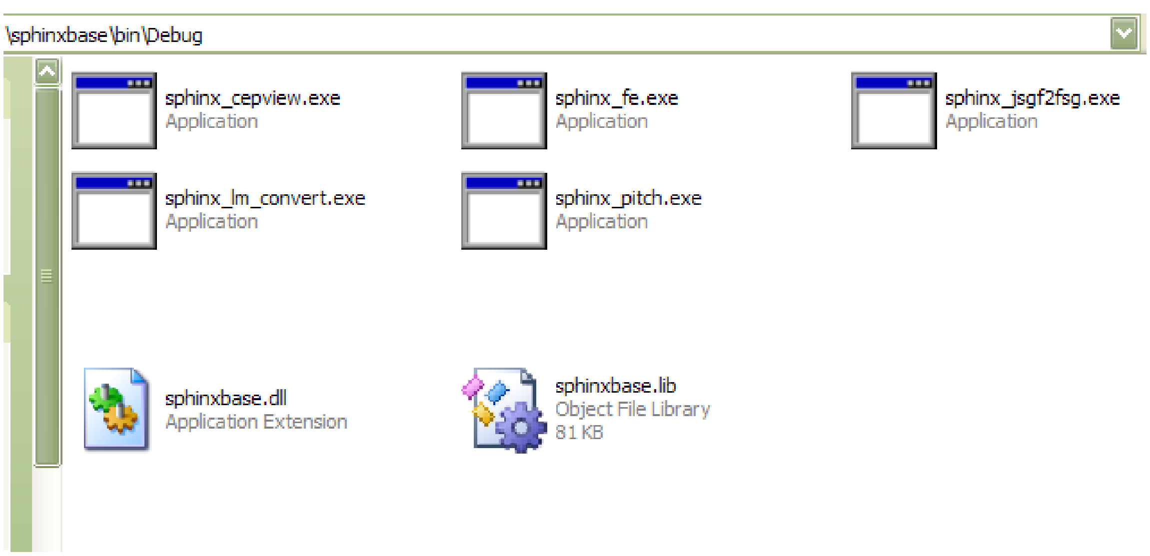


Figure 2 Executables and other files in sphinxbase

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.

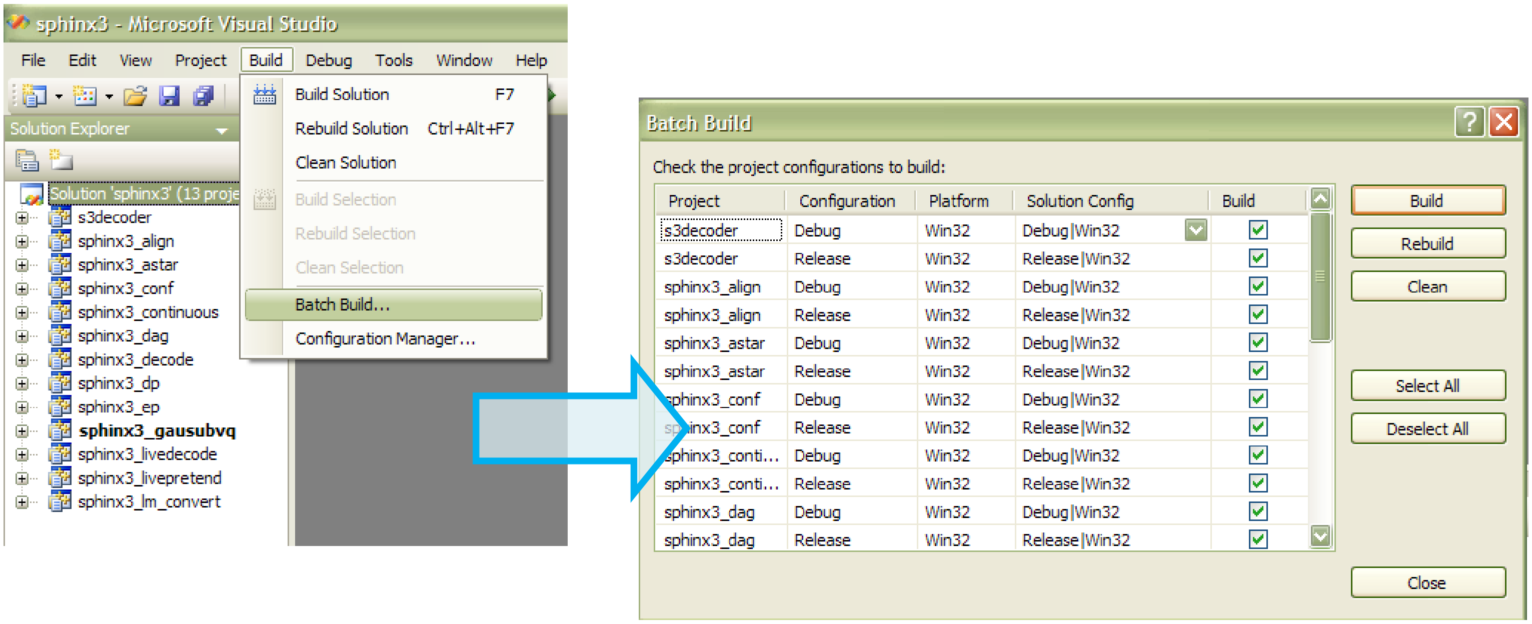


Figure 3 - Building Sphinx3

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**

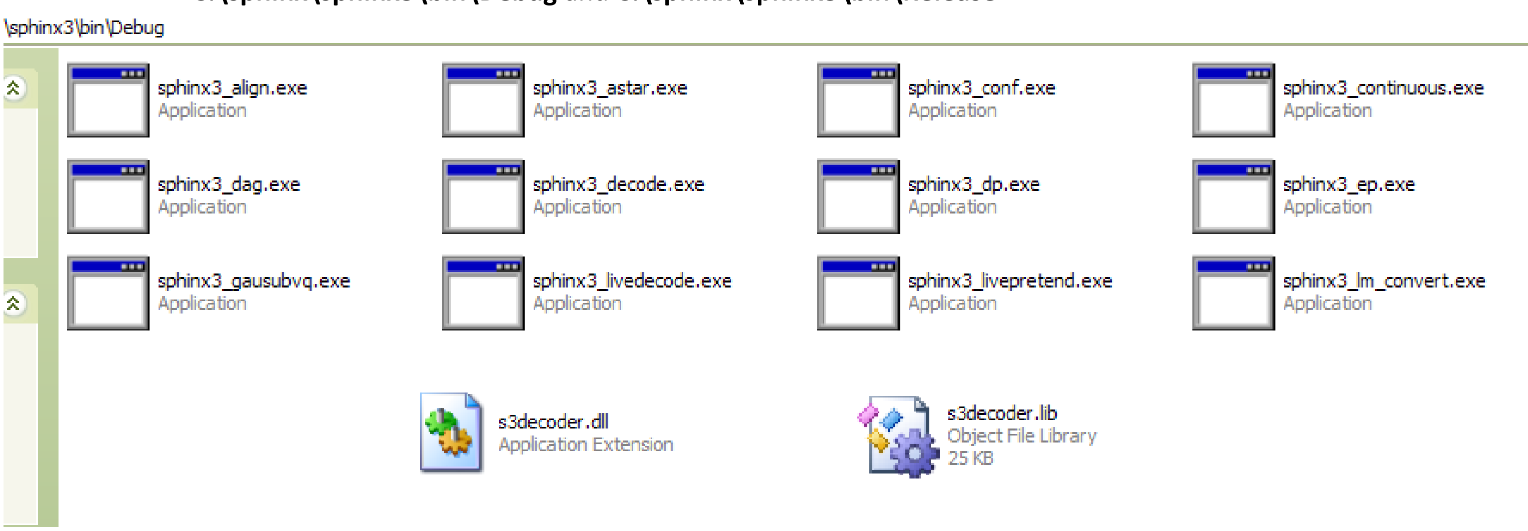


Figure 4 - Executables and other files in Sphinx3

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.)

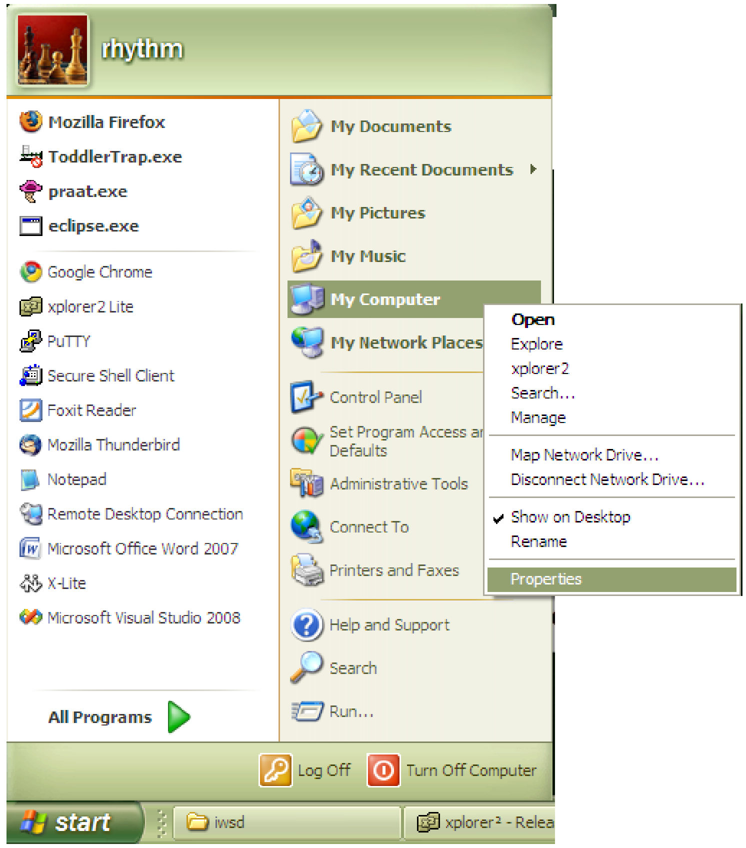

rhythm
Mozilla Firefox
ToddlerTrap.exe
praat.exe
eclipse.exe
Google Chrome
xplorer2 Lite
PuTTY
Secure Shell Client
Foxit Reader
Mozilla Thunderbird
Notepad
Remote Desktop Connection
Microsoft Office Word 2007
X-Lite
Microsoft Visual Studio 2008
All Programs
My Documents
My Recent Documents
My Pictures
My Music
My Computer
My Network Places
Control Panel
Set Program Access ar
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

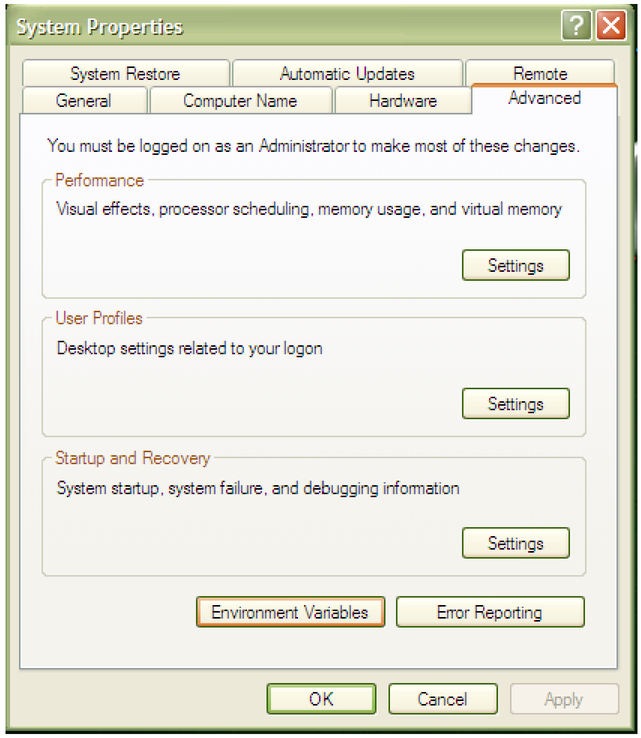


Figure 6

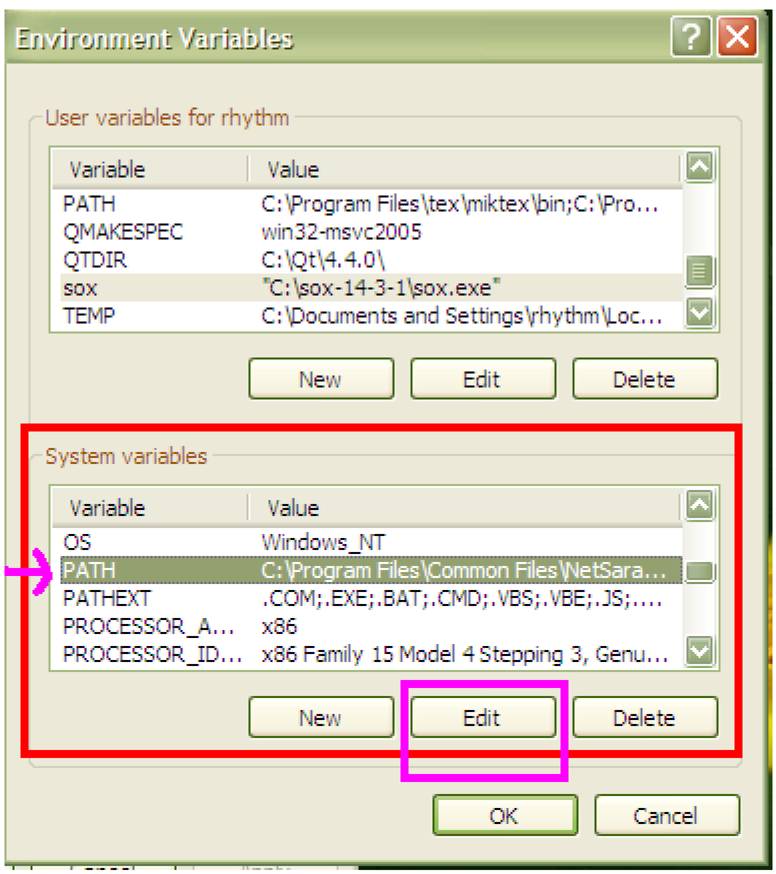


Figure 7

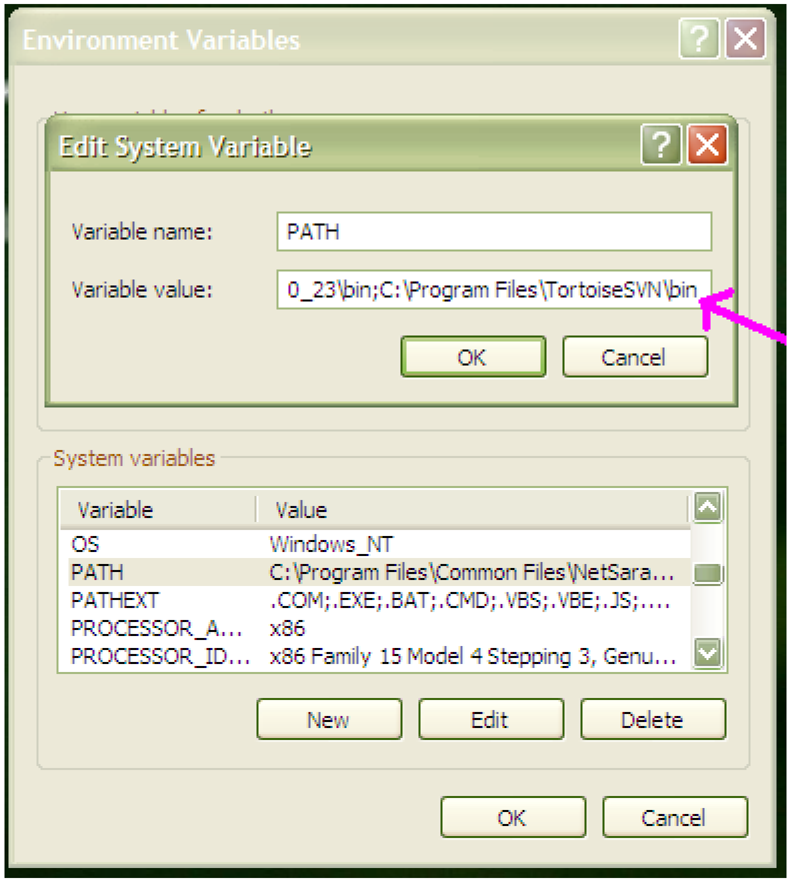

Environment Variables
Edit System Variable
Variable name:
PATH
Variable value:
0_23\bin;C:\Program Files\TortoiseSVN\bin
OK
Cancel
System variables
Variable
Value
OS
Windows_NT
PATH
C:\Program Files\Common Files\NetSara...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
PROCESSOR_A...
x86
PROCESSOR_ID...
x86 Family 15 Model 4 Stepping 3, Genu...
New
Edit
Delete
OK
Cancel

Figure 8

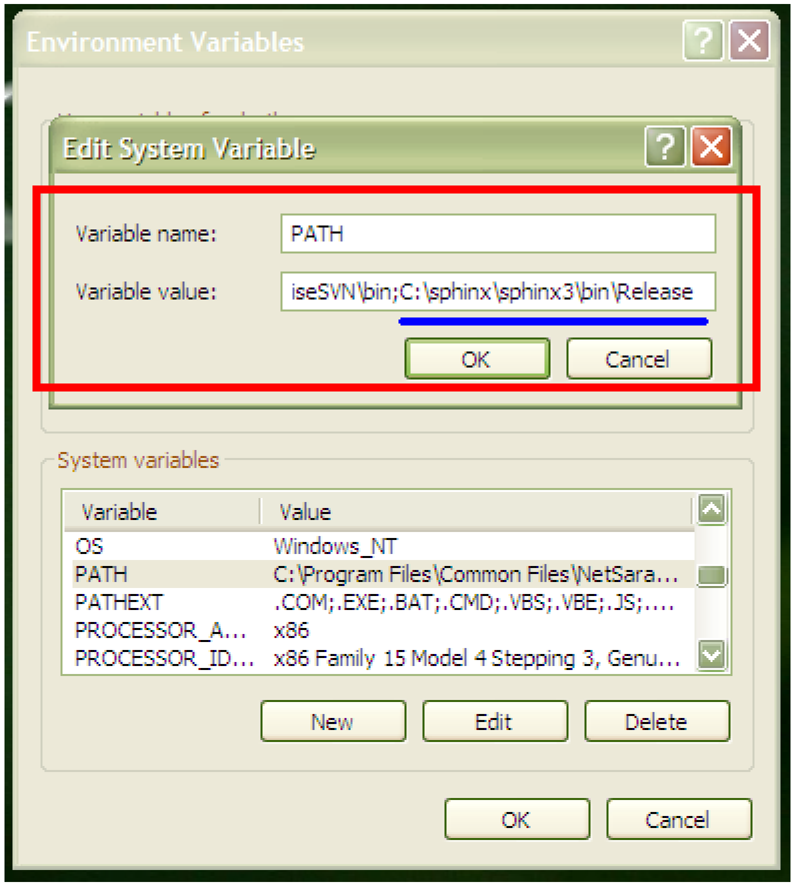


Figure 9

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]

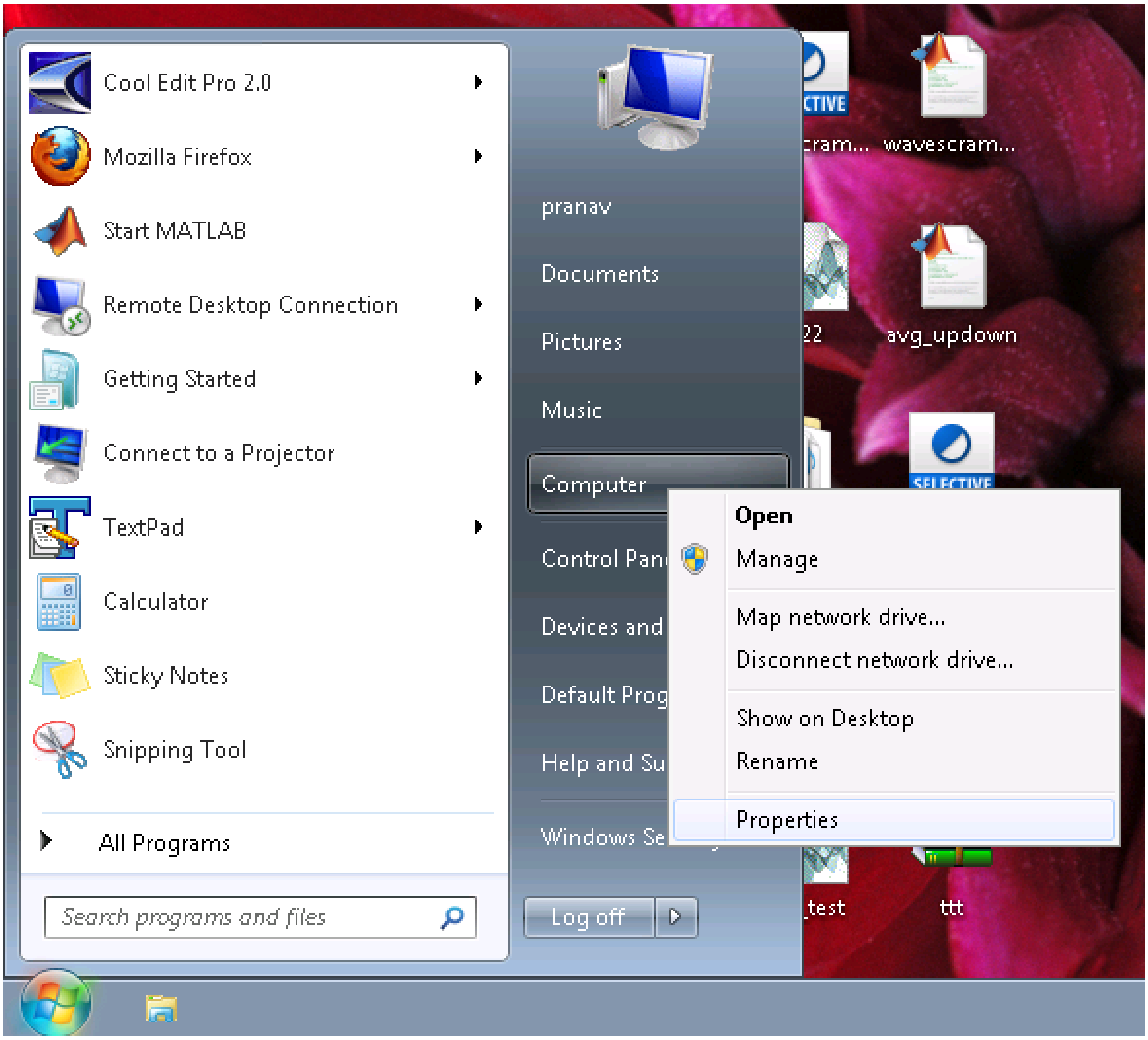

Cool Edit Pro 2.0
Mozilla Firefox
Start MATLAB
Remote Desktop Connection
Getting Started
Connect to a Projector
TextPad
Calculator
Sticky Notes
Snipping Tool
All Programs
Search programs and files
pranav
Documents
Pictures
Music
Computer
Control Pan
Devices and
Default Prog
Help and Su
Windows Se
Log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

Figure 10

Figure 11

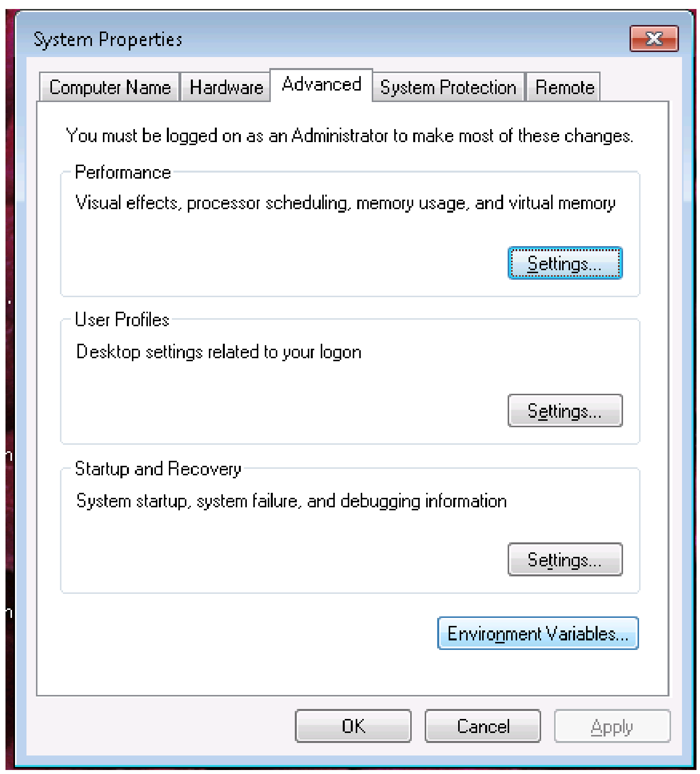


Figure 12

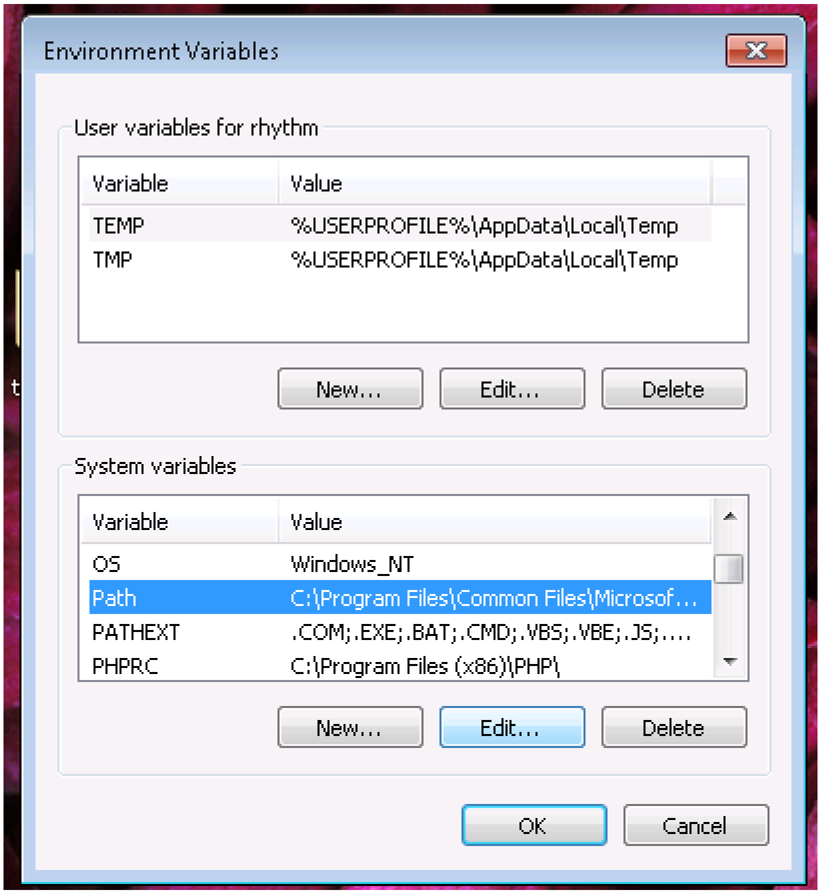


Figure 13

>> 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

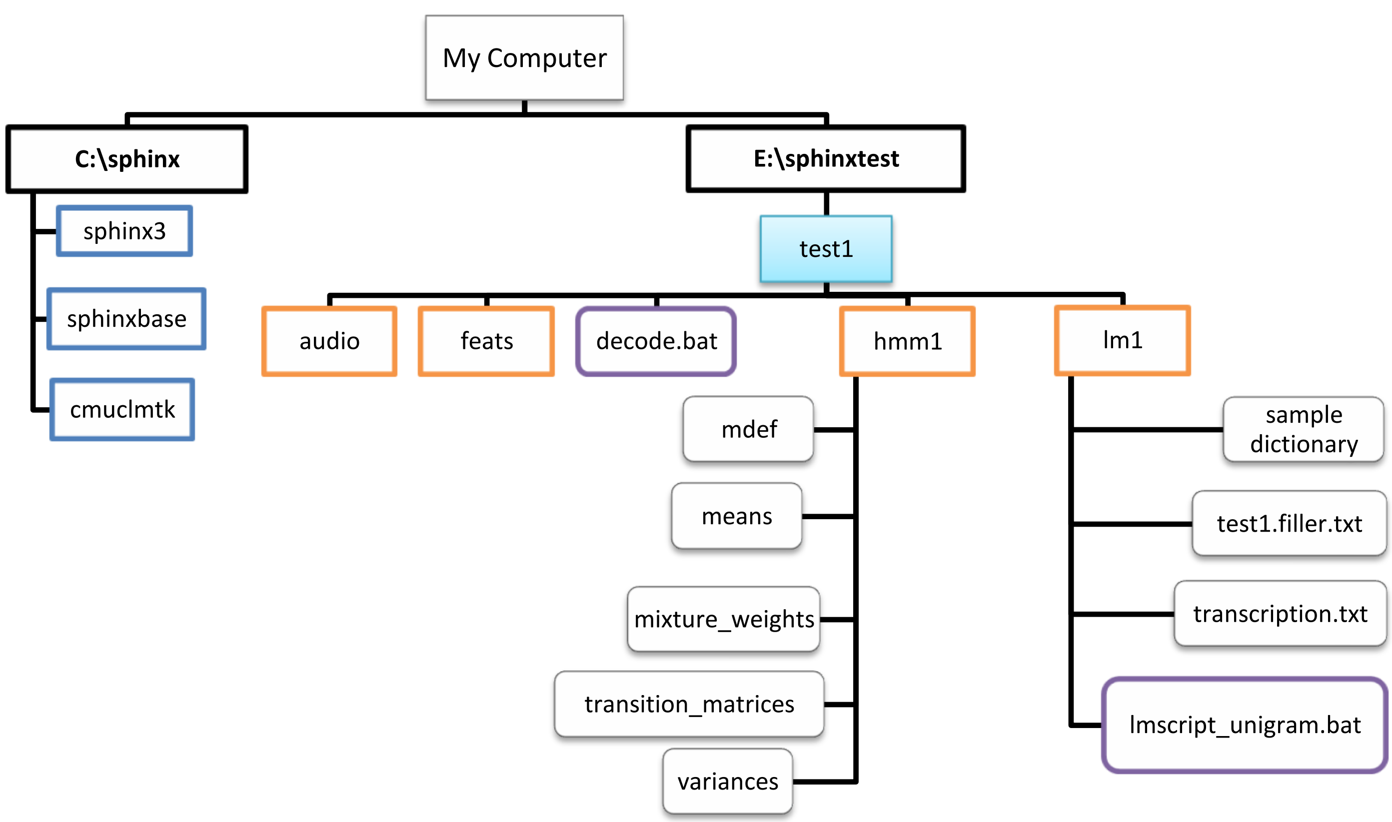
E:\sphinxtest

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, …, w_m) in the dictionary. P(W) is the probability of occurrence of this sequence. We can write P(W) as –

$$\mathrm{P}(\mathrm{W}) = \mathrm{P}(\mathrm{w}_1)\mathrm{P}(\mathrm{w}_2 \mid \mathrm{w}_1)\mathrm{P}(\mathrm{w}_3 \mid \mathrm{w}_1, \mathrm{w}_2) \ldots \mathrm{P}(\mathrm{w}_m \mid \mathrm{w}_1, \mathrm{w}_2, \ldots \mathrm{w}_{m-1}) = \sum_{i=1}^{m} P(\mathrm{w}_i \mid \Phi_i)$$

Φ_i is in some sense the *history* of the i^{th} word.

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

$$\mathrm{P}(\mathrm{W}) = \mathrm{P}(\mathrm{w}_m)\mathrm{P}(\mathrm{w}_{m-1} \mid \mathrm{w}_m)\mathrm{P}(\mathrm{w}_{m-2} \mid \mathrm{w}_m, \mathrm{w}_{m-1}) \ldots \mathrm{P}(\mathrm{w}_1 \mid \mathrm{w}_m, \mathrm{w}_{m-1}, \ldots \mathrm{w}_2) = \sum_{i=1}^{m} P(\mathrm{w}_i \mid \Phi'_i)$$

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

$$\mathrm{P}(\mathrm{W}) = \mathrm{P}(\mathrm{w}_1)\mathrm{P}(\mathrm{w}_2)\mathrm{P}(\mathrm{w}_3) \ldots \mathrm{P}(\mathrm{w}_m) = \sum_{i=1}^{m} P(\mathrm{w}_i)$$

So the occurrence of each word is independent of any other word. Further assume that each of $P(\mathrm{w}_i)$ 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).

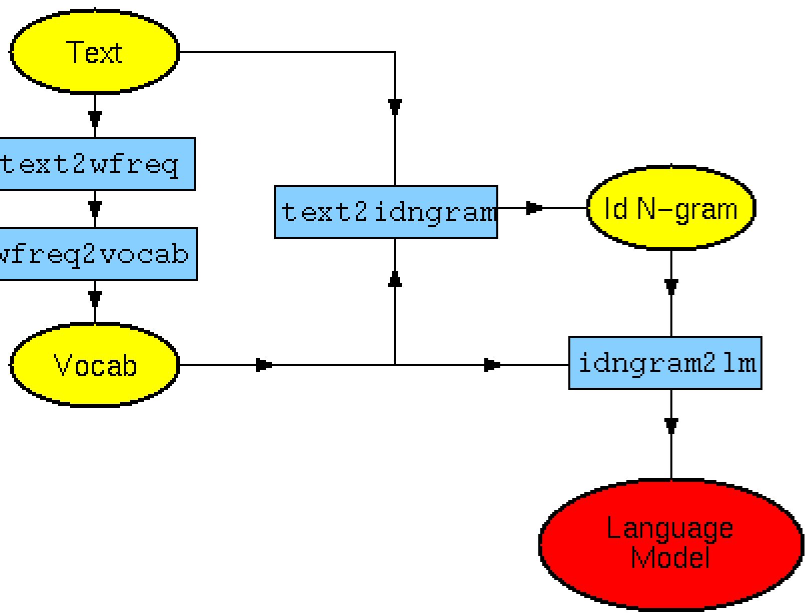


Figure 14 - Language model creation (from http://www.speech.cs.cmu.edu/SLM/toolkit_documentation.html#typical_use)

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.

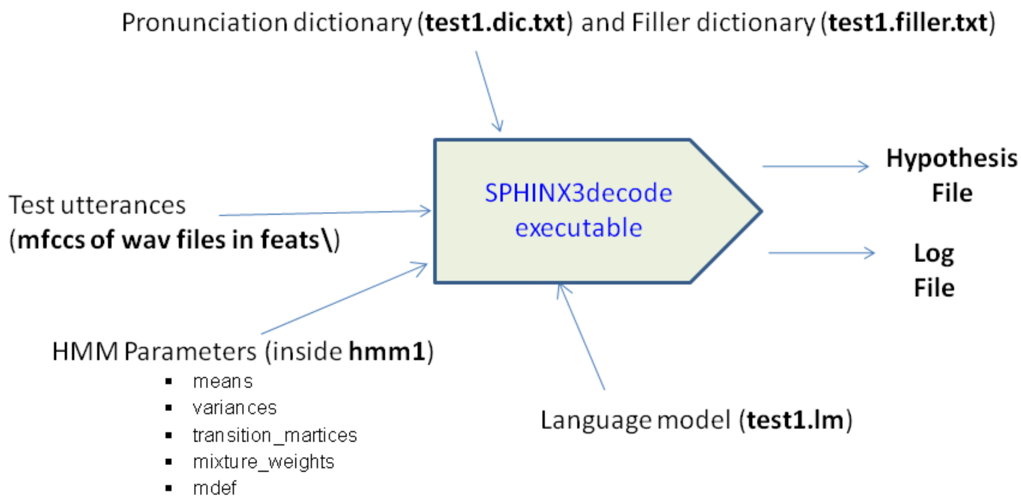


Figure 15 "Inputs and outputs" Sphinx3_decode

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
---