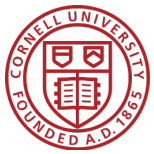
SPOKEN LANGUAGE ACCENT DETECTION

Probabilistic Accent Detection Using Hidden Markov Models

Tré Calhoun
La Vesha Parker
Andrew Vaslas
Nicolas Vera
Cornell University



Cornell University

CONTENTS

PREFACE

All information presented within this document represents our exploration of the HTK software. We make no guarantee of things things things

INTRODUCTION

This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction. This is the introduction.

PART I

HIDDEN MARKOV MODEL TOOLKIT SOFTWARE SUITE

CHAPTER 1

INSTALLATION OF HTK SOFTWARE

1.1 General Installation information

The website for HTK can be found here. The HTK developers require that you register for a username and password through their site before downloading their software. After registering, visit the downloads page and download the HTK source code (available as a tarball). It is also useful to download the HTKBook as a PDF (available on the downloads page below the software). If you do not wish to download the book, you can view the book online after registering.

1.2 Mac OS X

In order to install HTK for Mac OS X, you first need to make sure that you have Xcode developer tools and X11 installed.

What follows are the installation instructions taken *directly from the README in the root directory of the unziped htk/ directory*, save a bit of formatting. We do not claim this work, and repeat it here only for convenience.

1.2.0.1 Compiling & Installing HTK under UNIX/Linux, OS X or Cygwin

After unpacking the sources, cd to the htk directory.

There are now two ways to install HTK, the ”traditional” and the ”new”. Up to now HTK has always installed its tools as they were built, and installed them to a directory such as ”bin.linux” so that binaries for different architectures can be installed in a home directory say. If you want to install in this way, please add the option ”–enable-trad-htk” when you run configure.

The ”new” method installs by default into /usr/local/bin (equivalent to a configure option of ”–prefix=/usr/local”).

1. decide which of the above methods you wish to use
2. cd to htk, then run ./configure (with appropriate options, run ”./configure –help” if unsure). If you don’t want to build the programs in HLMTools add the –disable-hlmtools option.
3. make all
4. make install

Running ”make install” will install them. This step may need to be done as root, if you are not installing them in your home directory.

Notes for particular Unix variants:
Solaris: if ”make” isn’t installed you may need to add /opt/sfw/bin and /usr/ccs/bin to your path and run ”./configure MAKE=gmake” with any other options you require. Then run ”gmake” instead of ”make”, alternatively you can create a symbolic link called ”make” somewhere it your path to /opt/sfw/bin/gmake

1.3 Windows

Once again, what follows are the installation instructions taken *directly from the README in the root directory of the unziped htk/ directory*, save a bit of formatting. We do not claim this work, and repeat it here only for convenience.

1.3.0.2 Compiling & Installing HTK under Windows

Prerequisites:

- HTK has been verified to compile using Microsoft Visual Studio.
- For testing, you will require a Perl interpreter such as ActivePerl.
- You will need a tool such as 7-zip or winzip (commercial) for unpacking the HTK source code archive.
- It it is helpful if you have some familiarity with using the DOS command line interface, as you will need to interact with it in order to compile, install and run HTK.
- Ensure that your PATH contains:

```
C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin
```

Or if you are using older versions:

```
C:\Program Files\Microsoft Visual Studio\VC98\bin
```

Compilation:

1. Unpack the HTK sources using 7-zip.
2. Open a DOS command window: Click Start, select Run type cmd at the prompt and click OK.
3. cd into the directory in which you unpacked the sources.
4. cd into the htk directory. Type:

```
cd htk
```

5. Create a directory for the library and tools. Type:

```
mkdir bin.win32
```

6. Run VCVARS32 (it should be in your path, see prerequisites above) 7. Build the HTK Library, which provides the common functionality used by the HTK Tools. Enter the following commands:

```
cd HTKLib
nmake /f htk_htklib_nt.mkf all
cd ..
```

8. Build the HTK Tools

```
cd HTKTools
nmake /f htk_htktools_nt.mkf all
cd ..
cd HLMLib
nmake /f htk_hlmlib_nt.mkf all
cd ..
cd HLMTools
nmake /f htk_hlmtools_nt.mkf all
cd ..
```

Installation:

The HTK tools have now been built and are in the bin.win32 directory. You should add this directory to your PATH, so that you can run them easily from the command line in future.

CHAPTER 2

TRAINING AND TESTING CORPUS ACQUISITION

> Everyone has the right to life, liberty and security of person.
>
> —United Nations' Declaration of Human Rights [4]

2.1 The Nature of Our Data

2.2 The Online Speech/Corpora Archive and Analysis Resource

Northwestern University's Online Speech/Corpora Archive and Analysis Resource (OSCAAR) is a collection of speech recordings from speakers with different backgrounds, assembled from various datasets.

To request access to the data available through OSCAAR, you can submit a request for access to the OSCAAR collections. In our experience, requests are handled within 24-48 hours after being sent.

The dataset that we found most appropriate for our goal of accent detection and classification is the ALLSTAR dataset from the Speech and Communication Research Group at Northwestern University. The dataset is massive, and we found that a subset of samples fit our needs well.

Part of the dataset features recording of talkers from different backgrounds saying 20 sentences pulled from the Declaration of Human Rights in English. For our proof of concept, we used a subset of that portion of speakers reading Article 3 from the DHR: "Ev-

eryone has the right to life liberty and security of person." That subset of the data featured talkers with the following native tongues:

- Brazilian Portuguese
- English
- French
- German
- Hebrew
- Hindi
- Japanese
- Korean
- Mandarin Chinese
- Persian (Farsi)
- Russian
- Spanish
- Turkish
- Vietnamese

CHAPTER 3

TRAINING CORPUS WITH HTK

3.1 Record or Input Sound Files

Having acquired the sound clips from OSCAAR, we needed to select two distinct native tongues. Native English and native Spanish speakers were selected since these subsets had what we felt was a sufficient amount of data for the purposes of this project; some of the others only had a few samples and having more data is conducive to the training of the HMM.

Some of our data was set aside for testing. In data/train/ and data/test/, the .wav audio clips themselves are stored in wav/ and their respective .mfcc and .lab files were stored in mfcc/ and lab/. The free audio editor Audacity was used to crop the .wav files so that only the word ”security” could be heard. Audacity allowed us to generate silent audio surrounding the cropped clips.

3.2 Labeling the Sound Files

HLab allowed us to label the boundaries between words and the silence around them in each .wav file. With a beginning silence, the spoken word, and the ending silence labeled, a .lab file for each clip was created. The .lab files are merely text files marking the start and end sample times for each of these labeled sections:

```
data/train/lab/english_f1_security.lab
20408 4239909 sil
```

```
4342857 9941497 security_english
9982766 13996825 sil
```

CHAPTER 4

CODING THE DATA

4.1 Mel Frequency Cepstral Coefficients

Here we describe what a MFCC is and its usefulness to us.

4.2 Obtaining .mfcc Files

HLab also produces a .sig signal file upon opening the audio clip. Once saved, these files were converted into .mfcc files using HCopy. The .sig files themselves cannot be analyzed. The .mfcc files, which each contain a set of vector representations of the sound signal, can be analyzed. Each 25ms segment is represented by a vector of acoustical coefficients, which provides a description of that segment's spectral properties. HCopy required a configuration file to give values to its parameters:

4.2.1 Configuration File

```
#analysis.conf
SOURCEFORMAT = WAV    # Gives the format of the speech files
TARGETKIND = MFCC_0_D_A    # Identifier of the coefficients to use

# Unit = 0.1 micro-second :
WINDOWSIZE = 250000.0    # = 25 ms = length of a time frame
```

```
TARGETRATE = 100000.0    # = 10 ms = frame periodicity

NUMCEPS = 12     # Number of MFCC coeffs (here from c1 to c12)
USEHAMMING = T    # Use of Hamming function for windowing frames
PREEMCOEF = 0.97     # Pre-emphasis coefficient
NUMCHANS = 26     # Number of filterbank channels
CEPLIFTER = 22     # Length of cepstral liftering
```

4.2.2 The Creation of targetlist.txt

The file trainlist.txt gives the name and directory of each waveform file and their respective target coefficient files:

```
data/train/wav/english_f1_security.wav data/train/mfcc/english_f1_security.mfcc
data/train/wav/english_f2_security.wav data/train/mfcc/english_f2_security.mfcc

(and so on)
```

4.3 Command Line Actions

To execute this conversion, the following command was used:

HCopy -A -D -C analysis.conf -S trainlist.txt

CHAPTER 5

SETTING PARAMETERS FOR THE HIDDEN MARKOV MODEL

Multiple things should happen here:

1. Explain what an HMM is and what it is useful for
2. Explain particularly why it works for what we are doing
3. Describe the input parameters to a hidden markov model
4. Explain why we made any changes to what the original tutorial had/any issues we encountered (i.e. errors being raised when we tried to have too many states due to not having enough training examples for all those states)

I have some sample sections below following the list above:

5.1 What is a Hidden Markov Model?

Here is some sample text.

5.2 HMMs and Accent Detection

Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum

5.3 Input Parameters to HMMs

Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum Lorem ipsum

5.4 Justification for our Modifications

5.5 Command Line Actions

CHAPTER 6

DEFINING THE GRAMMAR OF YOUR NETWORK

CHAPTER 7

TESTING WITH NEW SAMPLES

Corresponds to the Recognition chapter (Moreau ch. 7)

7.1 Command Line Actions

PART II

DATA VISUALIZATION

PART III

ERROR HANDLING, SOFTWARE USED AND RESOURCES

APPENDIX A

ERROR HANDLING

Errors that crop up when using HTK can seem confusing initially, but they can be debugged with a bit of critical thinking and some hints from outside sources. We found the following sources helpful when debugging:

- Ohio State University: Understanding HTK Error Messages
- Columbia University: Summary of Errors by Tool and Module

During our time with the HTK software, we experienced a subset of the error codes that we could experience. If you encounter theses error codes yourself, there is no guarantee that our error resolutions will also work for you, but hopefully they will help with debugging seeing

APPENDIX B

SOFTWARE USED

We have provided information about our use of various software (particularly the various toolkits available through the HTK software) throughout this guide, so we will merely provide a summary of what

1. Audacity: Useful for processing audio files. Used to splice .wav files to extract specific words from .wav files. Also useful for generating segments of silence to make differentiating between SIL labels and spoken words easier.

2. HTK: Maybe even list each of the things we used under HTK & why, i.e. HSLab for labeling, HParse for whatever

   (a) **HSLab**
   Provides a graphical user interface for the labeling of sound files. Accepts waveform files (i.e. .sig files recorded directly in HSLab and .wav files that users can pre-record). The default expected filetype is .sig, so if you plan to use a different file type, be sure to include a configuration file (i.e. our analysis.conf) that specifies the SOURCE FORMAT.

   Sample command line invocation:

   ```
   HSLab -C <config_file.conf> <sound_file.ext>
   ```

   (b) **HCopy**
   The primary use of HCopy is to copy and manipulate speech files. Another use for

HCopy (and our particular use here) is to convert waveform data to Mel Frequency Cepstral Coefficients. HCopy accepts pairs of source specifications for .lab files and destination specifications for the .mfcc files that HCopy will generate for each of the .lab files.

Sample command line invocation:

```
HCopy -C <config_file.conf>  -S testlist.txt
```

(c) **Hinit**
Sample command line invocation:

```
HInit -A -T 1 -S hinit_trainlist.txt -M model/hmm0 -H model/proto/hmm_security_engli
```

(d) **HRest**
Sample command line invocation:

```
HRest -A -T 1 -S hinit_trainlist.txt -M model/hmm1 -H model/hmm0/hmm_security_englis
```

(e) **HParse**
Sample command line invocation:

```
HParse -A -T 1   def/gram.txt   net.slf
```

(f) **HVite**
Sample command line invocation:

```
HVite -A -D -T 1  -H hmmsdef.mmf  -i reco.mlf  -w net.slf
```

Global useful flags:

(a) f

APPENDIX C

REFERENCES

REFERENCES

[1] Random People, "Hidden Markov Model,"(2014).

[2] Random People, 'HTK Basic Tutorial'(2014).

[3] Random People, 'HCopy Config File'(2014).

[4] UN General Assembly, *Universal Declaration of Human Rights*, 10 December 1948, 217 A (III), available at: http://www.refworld.org/docid/3ae6b3712c.html