AFRL-RI-RS-TR-2012-294

OPEN-SOURCE MULTI-LANGUAGE AUDIO DATABASE FOR SPOKEN LANGUAGE PROCESSING APPLICATIONS

BINGHAMTON UNIVERSITY

DECEMBER 2012

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2012-294 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

STANLEY J. WENNDT
Work Unit Manager

/ S /

WARREN H. DEBANY, JR.
Technical Advisor, Information
Exploitation & Operations Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
DECEMBER 2012	FINAL TECHNICAL REPORT	APR 2010 – APR 2012

4. TITLE AND SUBTITLE

OPEN-SOURCE MULTI-LANGUAGE AUDIO DATABASE FOR SPOKEN LANGUAGE PROCESSING APPLICATIONS

5a. CONTRACT NUMBER
FA8750-10-2-0160

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
65502F

6. AUTHOR(S)

Stephen Zahorian

5d. PROJECT NUMBER
3480

5e. TASK NUMBER
BA

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical and Computer Engineering
Binghamton University
Binghamton, NY 13902-6000

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RIGC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-294

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report gives a detailed summary of research work completed under Air Force Research Laboratory (AFRL) grant 53925, over the time period (April 12, 2010 – April 10, 2012). There are two main aspects of the work completed. First was the collection and annotation of a large open source data base of speech passages from web sites such as You Tube. 300 passages were collected in each of three languages—English, Mandarin, and Russian. Approximately 30 hours of speech were collected for each language. Each passage has been carefully transcribed at the phrasal level by human listeners. Each passage was originally transcribed and then checked and the transcription edited as needed by at least two additional native language listeners. The English and Mandarin were then forced aligned and labeled at the phonetic level using a combination of manual and automatic methods. The Russian passages have not yet been marked at the phonetic level. Another phase of the work was to explore several algorithmic methods for improving automatic speech recognition (ASR) for this intelligible but challenging data base. Note that the body of the report has four main sections plus appendices which introduce, describe, and summarize a portion of the work.

15. SUBJECT TERMS
Open-source videos, speech recognition, forced alignment

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**STANLEY J. WENNDT**
U	U	U	UU	79	19b. TELEPONE NUMBER (*Include area code)* **NA**

Table of Contents

Section **Page**

LIST OF TABLES

Tables **Page**

.

LIST OF FIGURES

Figure **Page**

1. SUMMARY

This report gives a detailed summary of research work completed under Air Force Research Laboratory (AFRL) grant 53925, over the time period (April 12, 2010 – April 10, 2012). There are two main aspects of the work completed. First was the collection and annotation of a large open source data base of speech passages from web sites such as You Tube. 300 passages were collected in each of three languages—English, Mandarin, and Russian. Approximately 30 hours of speech were collected for each language. Each passage has been carefully transcribed at the phrasal level by human listeners. Each passage was originally transcribed and then checked and the transcription edited as needed by at least two additional native language listeners. The English and Mandarin were then forced aligned and labeled at the phonetic level using a combination of manual and automatic methods. The Russian passages have not yet been marked at the phonetic level. Another phase of the work was to explore several algorithmic methods for improving automatic speech recognition (ASR) for this intelligible but challenging data base. Note that the body of the report has four main sections plus appendices which introduce, describe, and summarize a portion of the work.

2. INTRODUCTION

The primary sections of this report are

1. Database Development: This section describes the database in some details, various issues involved, and the transcription/labeling process.
2. Time Frequency Resolution Issues: This section describes some of the work done to improve automatic speech recognition accuracy using time-frequency resolution which varies as a function of frequency. At low frequencies, frequency resolution is high and time resolution low. At high frequencies, frequency resolution is low and time resolution is high.
3. Non Linear Amplitude Scaling: This section describes uses of a spectral nonlinearity, after log scaling, to improve noise robustness for ASR. It is shown that certain types of "simple" nonlinearities can improve ASR accuracy for the condition of mismatched training and test data.
4. New Approach to the Hidden Markov Model Decoding Paradigm: An algorithm is developed which reformulates Hidden Markov Model decoding which simplified the decoding and allows neural networks to be directly used in the decoding process. For the limited conditions tested, the accuracy is improved with this new approach.
5. Appendix A: As a result of this work, four conference presentations were given and one conference paper was published. The abstracts for the presentations and the conference paper are included.
6. Appendix B.1: A detailed description of the database, such as the file types and directory structure is given.

7. Appendix B.2: The phonetic alignment procedure, used for labeling at the phone level, is described.
8. Appendix B.3. A user's guide for the forced alignment software is given.
9. List of Symbols, Abbreviations and Acronyms: A list of acronyms used in this report.

Note that the each section in the main body of the report also has its own introduction.

3. METHODS, ASSUMPTIONS AND PROCEDURES

3.1. Database Development

3.1.1. Summary

Over the past few decades, research in automatic speech recognition and automatic speaker recognition has been greatly facilitated by the sharing of large annotated speech databases such as those distributed by the Linguistic Data Consortium (LDC). Open sources, particularly web sites such as YouTube, contain vast and varied speech recordings in a variety of languages. However, these "open sources" for speech data are largely untapped as resources for speech research. In this paper, a project to collect, organize, and annotate a large group of this speech data is described. The data consists of approximately 30 hours of speech in each of three languages, English, Mandarin Chinese, and Russian. Each of 900 recordings has been orthographically transcribed at the sentence/phrase level by human listeners. Some of the issues related to working with this low quality, varied, noisy speech data in three languages are described.

3.1.2. Introduction

The need for large well-labeled databases for spoken language processing is well known. "There is no data like more data." is a comment made by MIT speech researcher Victor Zue at a speech recognition workshop in the 1980s. Despite the large number and vast sizes of speech databases developed since the 1980s, the comment by Victor Zue still rings true [1].

One of the first large databases developed for speech research was the TIMIT acoustic-phonetic continuous speech corpus. With joint efforts from Texas Instruments, SR International and MIT, TIMIT was published by the LDC in 1993. This database contains recordings of 630 speakers, each reading 10 sentences, in "studio" conditions. The data was then manually labeled with starting and ending points for each phone in each sentence. Even today, TIMIT is one of the most widely used speech corpora for phonetic level speech research.

However, the 5,040 sentences in TIMIT (the SA sentences are often removed), with a typical sentence duration of 5 seconds, only provide about 7 hours of total speech, which is insufficient for many recognition tasks. Since TIMIT, many other speech databases have been collected, transcribed, catalogued, and distributed by the LDC. For example, the English Broadcast News

Transcripts (HUB4) [2] database was launched by the LDC in 1996 and now contains approximately 100 hours of broadcast news and has all speech manually transcribed at the phrasal level, using the Rich Transcription (RT-04S) Evaluation Data guidelines developed by the National Institute of Standard and Technology (NIST) in 2004 [3]. In recent years, LDC announced their plan for developing a new speech database. The DARPA GALE program [4] is a very large speech database project with the goal of collecting speech in multiple languages from global broadcast news. In 2009 the LDC [5] reported that 4,000 hours of Arabic broadcast has been collected and 2,400 hours were selected for transcribing.

Currently, public video sharing websites such as YouTube are booming because sharing homemade videos has become very easy and more popular. About 65,000 videos have been uploaded daily since 2006, and this number continuously increases [6]. This seemingly "infinite" number of videos found on the web can provide a vast collection of speech data for speech research, and the topics and speaking styles corresponding to these collections are much more varied than those found in broadcast news. To tap into this large resource, an "open source multi-language speech database" project was developed and is described in this report.

3.1.3. Structure of the database

The goals

The database was developed with collections from three different languages: English, Mandarin Chinese, and Russian. The intent was to collect about 30 hours of speech in each language, consisting of 300 videos per language, with videos averaging about 7 minutes in duration. The intent was also to collect three videos from each of 100 speakers per language, with the three recordings from each speaker originally spoken on different days and under different recording conditions. Another goal was to have approximately an equal number of male and female speakers. As is discussed later, most but not all, of these guidelines were met. The only firm guidelines were that the audio portion of each video be of sufficient quality to be "reasonably" intelligible by a "typical" native speaker of the language, that there not be constant background noise (i.e., not have background music throughout the entire passage), and that no single passage be shorter than 1.5 minutes or longer than 16 minutes in duration. Since many videos were in fact longer than 16 minutes, a "stand-alone" primarily speech portion of the video was extracted (using Xilisoft Video converter) for the database.

Video download and post-preparation

The first step in this development was to identify, download, and store audio/video clips from public video sharing websites. All videos were downloaded in the highest quality format that the sharing sites supported and then stored in a standard format. Table 1 shows the typical sites used for each language, the download tools used, and the original file formats.

For those videos in a format other than MP4, further processing was done, so that all videos were saved as MP4 files. The step was done by Xilisoft video converter, too. A copy of each video in

its original format was also kept. Then all files were designated with a specific name based on the language, the gender of the speaker, the initials of the speaker, and the category of recording. The category of recording comprised "formal presentation" and "casual conversation" which refer to the conditions under which the recording was made. Table 2 shows some criteria for making a decision on which category a video clip belonged to, although the decisions were somewhat subjective.

Table 1. ***Typical video sharing websites and tools for downloading***

Language	Typical Webs	Download Tools	Original Format
English	Youtube.com	www.savevid.com	MP4
Chinese	Youku.com	www.flvcd.com	FLV
Russian	Rutube.ru	www.savevid.com	MP4

Table 2. ***Some criteria for separating "Formal Presentation" and "Casual Conversation"***

	Formal Presentation	Casual Conversation
Speech type	Speech prepared in advance	Casual talk, semi-spontaneous
Noise/Interruptions in recording environment	Quiet and not much noise	More noise and even distortion effects
Slang/Disfluencies	Very little	Usually a lot
Background music	None	Maybe a little

A standard file name consists of 5 parts, chosen to make it easier to sort and organize the videos. Figure 1 illustrates a file name given to a Chinese language video sample, which is categorized as a "casual conversation," spoken by a male speaker who has initials "WX." The detailed description and possible options for each part as well as their abbreviations in file name are given in Table 3.

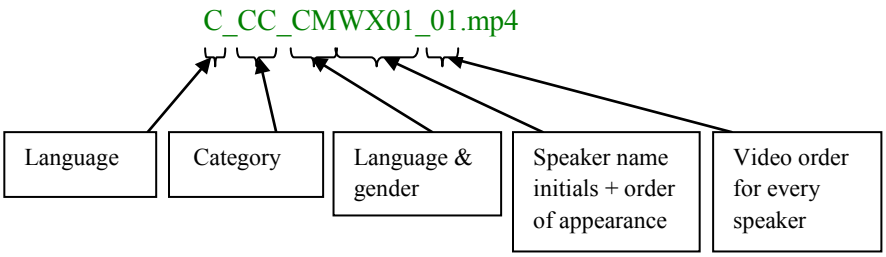

Figure 1. ***Illustration of naming conventions for video files***

Table 3. ***Filename notation and descriptions***

	Description	***In filename***
Language	English	E
	Mandarin Chinese	C
	Russian	R
Recording Category	Formal Presentation	FP
	Casual Conversation	CC
Gender	English/Chinese/Russian	E/C/R
	Male/Female	M/F
Speaker Name Initials	Initials of first / last name	
	Order of appearance in database	01, 02, 03…
Video Order	Indicates which video of each speaker	01, 02, 03…

3.1.4. Transcribing the database

An important aspect of this database is that all speech files were manually transcribed by human listeners. The reason for this was quite simple: given the relatively low quality, the wide variations in recording conditions, the presence of background noises, and the multiple languages, it seemed very doubtful that any automatic speech recognizer would be able to establish reliable "ground truth." By carefully listening to each sentence and reviewing by different listeners, the human transcriptions would provide the best orthographic transcriptions of this "ground truth." Also, the accurately transcribed sentences provides the best starting point for an automatic forced alignment process to create time labels for words and phonemes, thus better supporting phonetic recognition research. Therefore, properly selecting the tool and building the specifications for the transcribing work was necessary.

Transcribing tool

Transcriber, developed as a tool for assisting in creating the speech corpora in [7], was designed for manual segmentation and transcription of long duration broadcast news recordings, including annotation of speech turns, topics and acoustic conditions. With its embedded user-friendly graphical user interface, Transcriber allows listeners to perform tedious and complex operations such as modifying the time boundary of each speech segment, adding noise notations or indicate switching between speakers in a convenient way. Also, the output of Transcriber accurately records the time durations between segments, which provide support for the following automatic forced alignment process for detailed word and phonetic transcription. All these features made Transcriber extremely well-suited for the transcription task.

Other speech transcription tools considered are listed in Table 4. Although Transcriber does not have all the functions these other tools have, it does have the required features and there is no licensing fee; therefore, Transcriber 1.5.1 was used for this project.

Table 4. ***Other available transcription tools***

Tool	*Main Features*
XTrans	Multi-speakers tasks (Developed by LDC)
Transana	Link the transcription place to video
SoundIndex	Directly transcribe in XML file
WaveSurfer	Waveform display/analysis

Audio preparation

Transcriber only reads WAV files as its audio input. Therefore audio-only WAV files were extracted from the video MP4 files using a software tool called AOA audio extractor. Factors contributing to overall speech quality and intelligibility include: the background noise and recording conditions when videos were made, the loss by website compression tools for uploading files from users, another round of compression by the video download tool, and the final audio extraction. In order that the only significant degradation be due to the first two factors, high quality settings were used for the last two steps. Tables 5 and 6 list the quality setting for downloads and the quality setting for audio extraction, respectively.

Table 5. ***Video quality specifications***

Format	**MPEG-4**	
Video	Encoding	MPEG-4 AVC1 (H.264)
	Resolution	Original as on YouTube
Audio	Encoding	AAC
	Channels	2
	Sampling rate	44100Hz

Table 6. ***Audio quality specifications***

Format	**WAV**	
Audio	Encoding	PCM
	Channels	2
	Sampling rate	22050 Hz
	Bits/sample	16

Metadata and annotation

The annotation and metadata for transcribing this database was based on the format used for the LDC project GALE [8]. Unlike studio quality recorded read speech, there is a large amount of variability in transcribing web-collected speech. Main factors which possibly diminish the transcribing quality included the background noise/music, slang, and inserted words in different languages, other than the primary language of the speaker. These additional factors were annotated to the extent feasible.

When selecting videos, background music was considered permissible if it was not "too" high level and did not overlap with speech too often. Music, as well as other types of noise, was labeled in the transcription using the notation in Table 7. Noise labels differed depending on when the noise occurred relative to the speech. "Burst" noise/music occurred when there was no speech. "Overlap" noise/music occurs during speech. The most commonly used notation for noise was the "Others" notation, since, from listening, it was often quite difficult to accurately determine the source or type of the noise. Many speech signals also contained static noise which was present throughout the signal. This was labeled in a master file that describes each file.

Table 7. *Notations for noise and how it is annotated*

Noise Notation	Description/Example	Symbol/burst	Symbol/overlap
Music		[mu]	[mu-][-mu]
Applause		[a]	[a-][-a]
Laughing		[l]	[l-][-l]
Human	coughing, inhaling,	[h]	[h-] [-h]
Nature	wind, ocean...	[n]	[n-] [-n]
Vehicle	honks, car engine…	[v]	[v-][-v]
Animal	barking, birds…	[an]	[an-] [-an]
Office	telephone…	[o]	[o-] [-o]
Machinery	fan, construction...	[m]	[m-] [-m]
Others		[ot]	[ot-] [-ot]

Additionally, special events like silence segments and language transitions are potentially as detrimental as noise or other interference. While transcribing, the listeners labeled a "pause" (Table 8) as a speech break (silence) between 0.5 and 1 second. Silences longer than 1 second were labeled differently. For some run-on sentences which commonly occur in casual conversation, great care was taken with the time labels because speech often does not end abruptly, but rather gradually fades.

Table 8. *Notations for important events and how they are annotated*

Event	Description/Example	Symbol
Pause	Break btw/ .5 to 1 sec	[p]
Language Transition	Word(s) spoken in different language	Exp. [lang=English-] [-lang=English]

One common issue in spoken Mandarin Chinese is that people mix English words in Chinese sentences. (In contrast, Russian speakers often use English word variants.) Thus, labeling of languages transitions became necessary. Table 8 also shows the symbol for a language transition.

The issues related to slang, truncated words, incomplete pronunciation and other informality in spoken language are clearly addressed by the GALE standard. These issues include: contractions or ambiguous words should be transcribed as closely as possible to how they actually sound; truncated words are to be ended with a dash "-"; the notation "(())" is to be used to represent an unintelligible words; and tilde "~" is to be used when each letter of an acronym is spoken individually

3.1.5. Practical issues in transcribing

Common issues

The most difficult issue in the transcribing process of all three languages stemmed from the great range of qualities in the original video recordings. For example, some videos downloaded from YouTube had High-Definition quality, which has very high resolution for both audio and video. However, most videos were recorded with much lower quality for both video and audio. Some Chinese videos were recorded with defective equipment, resulting in distortion of the speech signal thus causing difficulties even for a human listener.

English

In English, speakers often fill pauses with fillers such as “um” or “hm,” briefly between words, or extensively to gain time for a next thought. Differences in pronunciation due to culture and accent promoted its own share of concerns; various accepted norms of speech (i.e. tomato), and the use of foreign language for brand names, proper nouns, descriptions, and verbs are used in conjunction with English speech. Furthermore, background noise or feedback from the medium used to record video was an additional factor.

Simplifying words and expressions through slang was very common; often times words that end in “-ing” are mispronounced and are transcribed as “-in.” For example “sleepin” for “sleeping.” And the use of “dope” and “hot” to express emotions or quality. The widespread use of internet acronyms such as “brb” and “lol” occurred occasionally in causal speech, implying the assimilation of today’s digital jargon in verbal communication.

Mandarin Chinese

Other than embedded English words, the transcription of Mandarin Chinese was done using standard Chinese characters for transcriptions. Unlike English, there are no slang or informal language (such as truncated words) related ambiguities in the transcriptions.

However, the Chinese language has a large number of dialects with a resulting large influence on how people pronounce Mandarin [9]. Accents among Mandarin speakers differ significantly. For example, the speakers from the northeast region of China confuse “s-” [s] and “sh-” [ʂ‘], and also there was no clear boundary between the nasal consonants “-ng” [ŋ] and “-n” [n] for south China speakers. In the development of the database described in this paper, the listeners always transcribed according to actual pronunciation, even if matching with its neighbor characters did not make linguistic sense.

Russian

In daily life, some words of the Russian language have a different pronunciation than defined in the dictionary. For example, “тыща” ['tɨɕɕa] is often pronounced “тысяча” ['tɨsjtɕa], “щас”

[ʂʂas] as “сейчас” [sej'tʂas], and “сёдня” ['sjdnja] as “сегодня” [se'gondnja]. Similar to Chinese and English transcriptions, all these ambiguous words were transcribed as pronounced.

Russians use English words directly sometimes. If the English word is clear with the correct pronunciation, the words are enclosed with English language tags. However, in some cases English words are misused (“Americanisms”): in such cases words are transcribed as they sound in Russian. Also, the Russian language includes some words with a Ukrainian pronunciation. Such words were marked with double parentheses.

3.1.6. Summary table of data collected

The goal of 300 videos for each language had been achieved, and roughly 30 total hours of speech for each database has been collected and transcribed, as summarized in Table 9. Over a period of about 9 months, which included about 3 weeks for data collection and approximately 8 months for transcribing, overall, 11 students were involved in the database development.

Table 9*. Summary of Database*

Language	*Speech Type*	*Gender*	*Total Number of Speakers*	*Total Number of Recordings*	*Total Time*
English	Formal	Male	88	124	14 hrs
	Formal	Female	21	26	2.6hrs
	Casual	Male	36	108	10.5 hrs
	Casual	Female	14	42	4 hrs
		Total	**159**	**300**	**31.1hrs**
Chinese	Formal	Male	59	136	11.1hrs
	Formal	Female	24	56	4.4 hrs
	Casual	Male	44	82	6.4 hrs
	Casual	Female	10	26	2 hrs
		Total	**137**	**300**	**23.9hrs**
Russian	Formal	Male	79	167	20.5hrs
	Formal	Female	26	42	4hrs
	Casual	Male	36	71	8.1hrs
	Casual	Female	17	20	2.2hrs
		Total	**158**	**300**	**34.8hrs**

3.1.7. Forced alignment for speech labeling

The goal of this database collection project is to make the database useful for speech processing research. In order to fulfill that goal, the database must provide accurate word-level and phone-level transcriptions, both with time marks. Although the speech data could be transcribed

manually by a well-trained phonetician, the manual method is a laborious and time-consuming task, which makes it impractical for a large amount of data. An efficient way of achieving this is to apply automatically derived forced alignment for this more detailed labeling and then use humans to finalize the labeling.

By using an ASR "tuned" for a single passage or group of passages, that is, given phonetic models and word models in terms of phonetic lattices, the audio can be accurately time aligned with word transcriptions. The recognizer can be configured to give a best time-aligned match between the audio and transcription, using all the probabilistic constraints imposed by the phonetic and word models. Furthermore, this recognizer can be made much more accurate by configuring it for each recording. For example, based on the human transcription of a recording, the vocabulary can be restricted to only the words in that recording, and the language model derived only from the word and word-pair frequencies in that passage. We are continuing work on several techniques for forced alignment on a subset of the English database and the outcome looks very promising. More details of the forced alignment are given in appendices B.2 and B.3. These techniques have been tuned for this particular task and database.

3.1.8. Conclusion

In this research project, a large database of English, Mandarin, and Russian was collected, formatted, organized, annotated, and given time-aligned orthographic transcriptions at the sentence/phrase level. Due to the variability, noisiness, and low speech quality, human listeners were employed for this transcription and annotation. Automatic speech recognition techniques have been developed and used to aid in the annotation process at a more fine-grained level. This database will be useful for both automatic speech recognition research and automatic speaker recognition research. The database is derived from open source public web sites; thus it is a sampling of an "infinite," widely accessed repository of speech.

3.2. Time Frequency Resolution Issues

3.2.1. Introduction

Speech information is generally assumed to be contained in a 2D time-frequency representation of speech spectra. This stems from a long history of using speech spectrograms, studies of the frequency analysis properties of the ear, etc. In the speech science field, the peaks in the spectrum corresponding to resonances of the vocal tract have long been considered to be most important features of the speech spectrum. However, in the field of ASR, the general approach is to extract features which represent the global spectral shape, and which mimic the amplitude and frequency selectivity of the human ear. The perceptual amplitude scale most typically used is logarithmic, and the perceptual frequency scale most typically used approximates a Bark or mel scale, often using a bilinear frequency transformation. In the ASR area, most typically, approximately 10-15 cepstral coefficients are computed from the log magnitude spectra. Cepstral coefficients are computed using a cosine transform of the log spectra, using a triangular filter

bank, with filters equally spaced on a bark-like scale. Since Hidden Markov Model (HMM) based ASR systems have been shown to benefit from including some cepstral trajectory information, the cepstral coefficients are often augmented with delta-cepstra and delta-delta cepstra terms, which approximate first and second order time derivatives of the cepstral coefficients. In fact, a de-facto standard front end for ASR systems is the use of 12 cepstral coefficients plus energy as base frame features, and then to augment each of these terms with delta and delta-delta terms. This results in a 39-dimensional feature vector. Over the past several years, researchers have introduced additional methods for representing spectral trajectory information, including RASTA [12] and the modulation spectrum [10] [11] [14].

3.2.2. Approach

Both Mel-Frequency Cepstral Coefficient (MFCC) features with associated delta and delta-delta terms, and our own previous features based on a warped Discrete Cosine Transform (DCT) of the log Fast Fourier Transform (FFT) spectra followed by a Discrete Cosine Series (DCS) expansion over time for each DCT [15] effectively use the same temporal resolution for all frequencies. In order to incorporate higher time resolution at high frequencies, as is possible with the lower frequency resolution, spectral/temporal features must first be computed by integrating over time (for each frequency) and then integrating the spectral/temporal features over frequency, or by using two-dimensional basis vectors over time and frequency. Figure 2 illustrates the type of desired time resolution over a segment interval for low and high frequencies. Note that the more "peaky" high frequency curve represents higher time resolution at the center of the interval, as compared to the less "peaky" low frequency curve. In the remainder of this paper, processing based on the two one-dimensional basis vector sets is referred to as DCTC1, whereas processing based on the two-dimensional basis vectors is referred to as DCTC2.

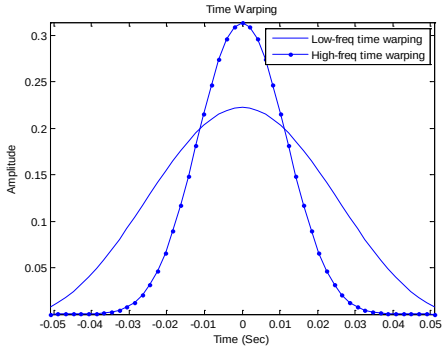

Figure 2. Illustration of desired time resolution at low frequencies and high frequencies.

The approach presented here is as follows. The DCTC2 features, Feat(i,j), are defined as :

$$Feat(i,j) = \int_{f=0}^{1} \Theta_i(f) \left(\int_{t=0}^{1} X(t,f) * \Psi_j(t,f) dt \right) df \quad (1)$$

Where *X(t,f)* is a (sliding) section of the time/frequency space (a block), with normalized ranges of (0,1) for both time and frequency,

$$\Theta_i(f) = \cos(i\pi g(f)) \frac{\partial g}{\partial f} \quad (2)$$

are modified cosine basis vectors over frequency, taking into account the frequency resolution properties of hearing, and

$$\Psi_j(t,f) = \cos(j\pi * hh(t,f)) \frac{\partial hh}{\partial t} \quad (3)$$

are two-dimensional basis vectors over time and frequency, with time resolution that depends on frequency. Note in equation 1 that the integration is over time first, and frequency second. *g(f)* and *h(f)* are warping functions over frequency and time respectively. *hh(t,f)* is a two-dimensional version of *h(f)*, or a family of *h* curves, with the degree of time warping depending on *f*. Typically *g(f)* is an approximation to the mel function, and *h(f)* is a function such as a Kaiser window, which results in more time resolution at the center of each block. Functionally, this method is equivalent to computing features based on two-dimensional basis vectors

$$Feat(i,j) = \int_{t=0}^{1} \int_{f=0}^{1} X(t', f') \cos(i\pi f') \sin(j\pi t') df' dt' \quad (4)$$

where *f'* is "perceptual" frequency, computed as *f' =g(f)*; and *t'* is "perceptual" time within a segment, computed as *t' =hh(t,f)*. The overall feature computation equation can be rewritten as:

$$Feat(i,j) = \int_{f=0}^{1} \int_{t=0}^{1} X(g(f), hh(t,f)) \Phi_{i,j}(t,f) dt df \quad (5)$$

In this equation *Φ(t,f)* incorporates both time and frequency warping. In the time dimension, the basis vectors are more sharply peaked at high frequencies than low frequencies, corresponding to the better time resolution at high frequencies. In the frequency dimension, the basis vectors are more sharply peaked at low frequencies than high frequencies, corresponding to the better frequency resolution at low frequencies. Figure 3 depicts the (1,1), (1,2), (2,1), and (2,2) basis vectors using mel warping for *g* (coefficient of .35) and Kaiser window (coefficient of 5) to control the time warping. These basis vectors have more "peakiness" at high frequency (over time) and low frequency (over frequency).

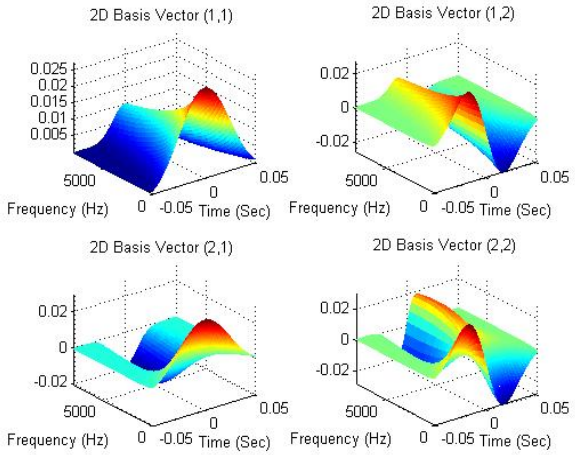

Figure 3. Low ordered time/frequency basis vectors illustrating non-uniform time/frequency resolution

To graphically illustrate the way these 2D basis vectors represent a speech spectrogram, original spectra and spectra estimated from the two-dimensional features *Feat(i,j)* are shown in figures 4 and 5. In Figure 4, the original spectrum (top panel) is obtained from 125 ms segments of sinusoids, spaced 500 Hz apart. The middle panel shows the rebuilt spectrum of the sinusoids using Discrete Cosine Transform Coefficient (DCTC)/ Discrete Cosine Series Coefficient (DCSC) features (13 DCTCs, 3 DCSCs, and uniform time resolution at all frequencies. As expected, the sinusoids are more smoothed in frequency at the high frequencies, but uniformly smoothed in time at both high and low frequencies. The bottom panel of figure 3 shows the rebuilt spectrogram using the 2D basis vectors. The frequency warping is based on the mel scale with a coefficient of .35. The time warping at low frequencies is based on a Kaiser window with beta of 5 at low frequencies, gradually increasing to a beta of approximately 15 at high frequencies (as in Figure 2). Note that the temporal transitions between sinusoids are much more rapid at high frequencies as compared to low frequencies.

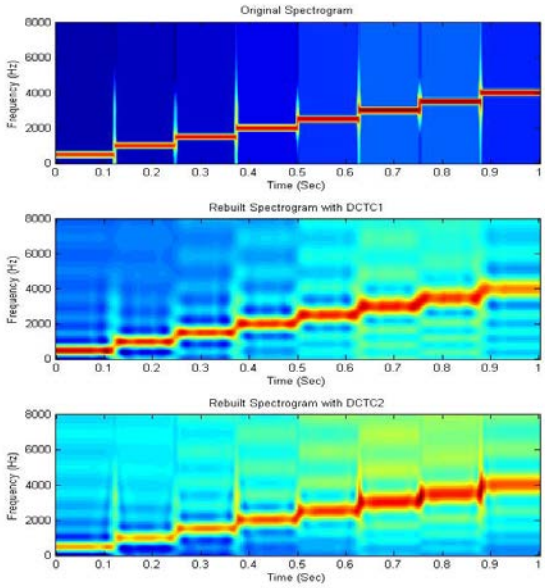

Figure 4. Spectrograms of sinusoids-original (top), reconstructed with DCTC1 method (middle) and DCTC2

The three panels of Figure 5 illustrate the original spectrogram (top panel), spectrogram rebuilt with time resolution the same at all frequencies (middle panel), and spectrogram rebuilt using 2D basis vectors (bottom panel), for approximately 1 second of a speech signal. As in Figure 4, in the bottom panel, the more rapid temporal events are better preserved for high frequencies than low frequencies.

3.2.3. Experimental evaluation

Experiments were conducted to evaluate the proposed method using the TIMIT database. The SA sentences were removed from the database, resulting in 3696 sentences from 462 speakers for training and 1344 sentences from 168 speakers for test. Experiments were also conducted with the telephone version of TIMIT, i.e. NTIMIT. All analysis parameters were identical for the TIMIT and NTIMIT evaluations, except for frequency range. For TIMIT, the frequency range was selected to be 50 to 7000 Hz, whereas for NTIMIT the frequency range for analysis was set at 300 to 3400 Hz.

The signal to noise ratio (SNR) was varied from no added noise (clean) to 0 dB, in steps of 10 dB (totally 5 noise conditions). A reduced 39 phone set as used in [13] was mapped down from the original TIMIT 62 phone set and used in the experiments.

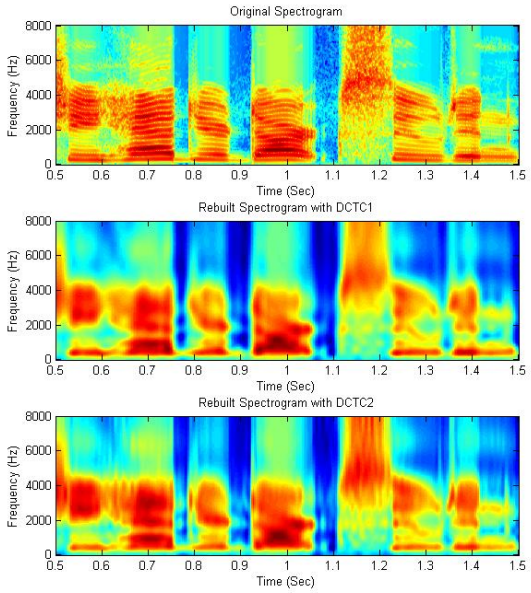

Figure 5. A segment of spectrograms for a speech signal (top), reconstructed with DCTC1 method (middle) and DCTC2 method (bottom).

Left-to-right 3-state Hidden Markov models with no skip were used and a total of 48 (eventually reduced to 39 phones) context independent monophone HMMs were created from the training data using the HTK toolbox (Ver3.4). The bigram phone information extracted from the training data was used as the language model. For all experiments with DCTC/DCSC features, a block spacing of 10 ms (125 blocks per second) was used. In an attempt to extract most information from the speech features tested, a large number of mixtures were used (32) to model each state with a diagonal covariance matrix.

The objective of the experiments was to compare phoneme recognition accuracy of control features (13 MFCCs with delta and acceleration terms, or 39 total terms), with DCTC/DCSC features computed with time resolution independent of frequency , and with the two-dimensional basis vectors which result in time resolution dependent on frequency. These three conditions are referred to as MFCC, DCTC1, and DCTC2 respectively. More details are given for each experimental condition.

Experiment 1: Control (MFCC)

The intent of this experiment was to establish a baseline for ASR phoneme accuracy using "conventional" features, and the identical HTK recognizer and database configuration as was used for the proposed features.

The MFCC features were computed directly with the HTK supplied front end, using the typical parameter settings for MFCCs as well as delta and acceleration terms. That is, 12 MFCCs plus energy with delta and acceleration terms, or 39 total terms were obtained every 10 ms at a frame length of 25 ms, with pre-emphasis coefficient of 0.97. For each phoneme, 3-state HMMs with 32 mixtures were used.

Recognition accuracies obtained with the TIMIT and NTIMIT databases at various SNRs were depicted in Figure 6. The accuracy ranges from 69% (clean TIMIT) to 35% (NTIMIT at 0 dB SNR).

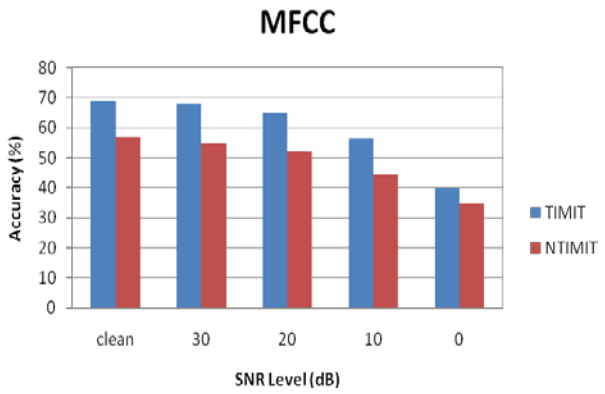

Figure 6. Phone accuracies with control MFCC features.

Experiment 2: DCTC1—DCTC spectral analysis followed by DCS spectral trajectory analysis

In this experiment, 13 DCTC coefficients were computed for each spectral frame, and each of these coefficients were represented with a 5 term Discrete Cosine Series (DCS) expansion over time (65 total features). The DCTC terms were computed with 10 ms frames, spaced 4 ms apart, with mel warping, using a coefficient of .45. The DCSC terms were computed using 50 frame overlapping blocks, with a 2 frame block spacing (that is, 200 ms blocks spaced 8 ms apart). The time warping function ("between" the low and high frequency curves in Figure 1) was a Kaiser window with beta value of 10. These parameters were adjusted empirically, to optimize for recognition accuracies.

The same configuration of HTK as for Experiment 1 was used--that is, three state models with 32 mixtures, and a bigram phonetic language model. Figure 6 shows that phoneme recognition accuracy for the TIMIT and NTIMIT databases at various SNRs, ranging from 72% (clean TIMIT) to 32% (NTIMIT at 0dB SNR). These results are approximately 3% higher, on average, than the results obtained with MFCC and delta features.

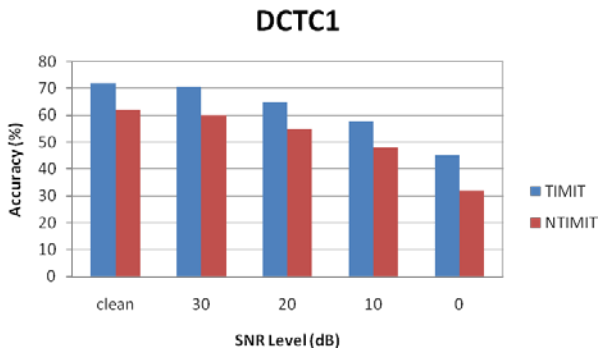

Figure 7. Phone accuracies with DCTC1 features

Experiment 3: DCTC2—features based on 2D spectral/temporal basis vectors

The goal of this experiment was to determine ASR accuracy possible using the 2D basis vector approach described in this paper. The number of parameters used (65) and parameter settings were similar to those mentioned for experiment 2. The effective Kaiser window beta was 5 for low frequencies, and 15 for high frequencies (as they are for the curves depicted in Figure 2). Again sliding 200 ms blocks, advanced by 8 ms, were used to compute the block based features. The same HMM configuration was used as for experiments 1 and 2.

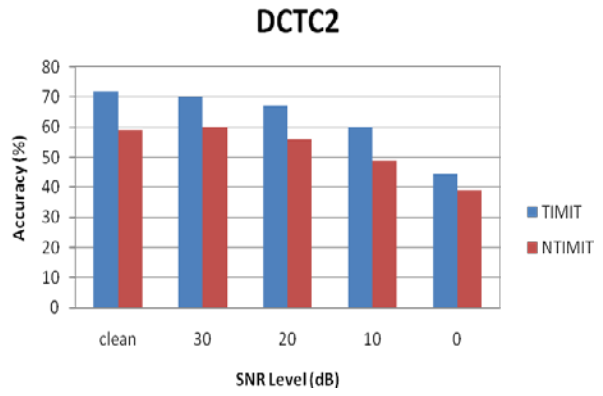

Figure 8. Phone accuracies with DCTC2 features

Figure 8 shows that phoneme recognition accuracy for the TIMIT and NTIMIT databases at various SNRs, ranging from 72% (clean TIMIT) to 39% (NTIMIT at 0dB SNR). These results are approximately 4% higher for TIMIT, on average, than the results obtained with MFCC and delta features, and also slightly better than for the DCTC1 features.

3.2.4. Discussion

The integrals mentioned in the algorithm are implemented as sums over selected time and frequency ranges. This "block" mode processing follows initial FFT spectral analysis. The nonlinear and frequency scales for the spectrum (equation 5) were obtained by interpolation. The combination of using very long block lengths (typically 50 frames), the interpolation, and the two-dimensional basis vectors, as opposed to using two sets of one dimensional basis vector does increase front end processing time by approximately an order of magnitude over MFCC and delta cepstra processing. More computationally efficient methods could be devised, if the features are shown to be beneficial.

3.2.5. Conclusions

A new method for computing spectral/temporal features for use in automatic speech recognition has been described. These features are computed so as to give time/frequency resolution that depends on position in time-frequency space. Although motivated by concepts from wavelet analysis, the algorithms described are developed in the acoustic feature domain, rather than the initial spectral analysis phase. Ideally, the initial spectral processing should also incorporate the non-uniform time/frequency resolution. Experimental results obtained to date are essentially equivalent to those obtained with features based on uniform time resolution. However, there remains considerable room for improvement using better matches to the time-frequency resolution characteristics of human hearing, and in using a more suitable spectral analyzer than FFT analysis.

3.3. Non Linear Amplitude Scaling

3.3.1. Introduction

ASR works well under clean conditions. This usually entails using clean speech and matching acoustical environments for the training and test data. However, when the ASR system has to operate under more adverse conditions recognition rates decrease quite dramatically [16].

In contrast to ASR, humans are able to comprehend speech despite adverse conditions. This has been the motivation for many techniques in the field of ASR. Some of these methods are derived from auditory models of human perception and other mammalian hearing systems. Auditory models are very complex and have been developed through various experiments which range from invasive procedures done on animals to otoacoustic emissions (OAEs) to psychoacoustics [17]. These models have led to the development of techniques such as MFCCs, RelAtive SpecTrA (RASTA) features, and Perceptual Linear Prediction (PLP) coefficients [18]. Auditory models can be used for ASR, but this comes at the cost of complexity [19].

Auditory models commonly have a compressive nonlinear stage that represents the processing done by the basilar membrane found in the cochlea of the ear [20,21]. Chiu and Stern [22,23] demonstrated that this critical step in hearing can be exploited using more traditional ASR techniques, but bypassing most of the more computationally intensive steps in a more complete auditory model. With their form of compressive nonlinearity they were able to improve the recognition rates for noisy speech using the CMU Sphinx-III system. Recognition rates were further improved by training the nonlinearity.

In this paper, we compare different compressive nonlinearities by integrating them with an ASR system that uses Zahorian's DCTC/DCSC acoustic features [15]. The compressive nonlinearities examined are derived from or similar to those in auditory models. We show how these nonlinearities shape the spectrum of the speech signal and how they affect recognition rates especially for noisy conditions.

3.3.2. Approach

The main idea behind these compressive nonlinearities is to limit the range of spectral magnitudes in such a manner that speech characteristics are enhanced and noise is suppressed. This can be done in many ways. The physiologically motivated way is to implement the compressive nonlinearity into more traditional ASR processing. A simple way is to limit the range of spectral magnitudes using a threshold, or "floor" value for spectral magnitudes.

To test the general idea of using compressive nonlinearities, noisy speech was made by adding white Gaussian noise at a desired signal to noise ratio (SNR). The speech used in the examples, and the speech used for the experimental results, were taken from the TIMIT database. The sentence used for the examples depicted in the figures in this paper is from a female speakers saying "She had your dark suit in greasy wash water all year."

Baseline Range Function

The baseline range function (BRF) is linear (in terms of log magnitude) except for a threshold:

$$y = \begin{cases} x, & x > c \\ c, & x \le c \end{cases} \qquad (6)$$

where the input x is the spectral magnitudes in dB and y is the spectral magnitude after compression. One way to define the threshold c is:

$$c = average\,(\max(x)) - t \quad (7)$$

$average\,(\max(x))$ is the average of the maximum values of spectral magnitudes for all frames of speech in a single utterance (one sentence) and t is a user defined value specified in dB. Therefore, this function limits the range of spectral magnitudes with the lower limit only dependent on the average spectral peaks of all frames of the utterance it is operating on. The value t is thus the limit for the dynamic range in dB for each utterance.

Figure 9 shows a spectrogram of an utterance of speech with no range limiting after the log compression, that is, the control condition. Figure 10 shows spectrogram with additive noise with a 10 dB SNR. Figures 11 shows a spectrogram of the 10dB SNR noisy speech, using the range limiting with t = 20 dB.

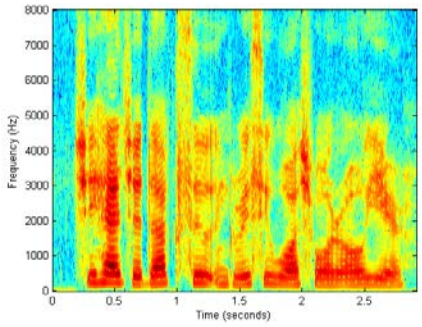

Figure 9. Spectrogram of clean speech with no range limiting.

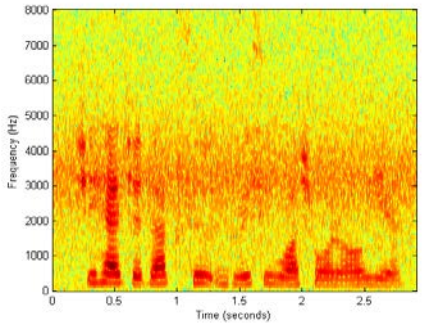

Figure 10. Spectrogram with additive Gaussian noise (10 dB SNR)

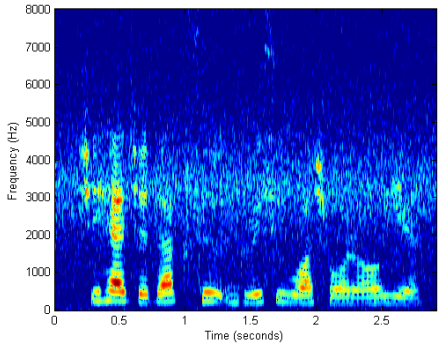

Figure 11. Spectrogram after applying range limiting *t* = 20dB

Compressive Nonlinearities

One group of compressive nonlinearities investigated in this work was derived from auditory models (Table 10). They were chosen for their general shape and lack of parameters tied to a specific auditory model. This ensures that these compressive nonlinearities can simply be incorporated in an automatic speech recognition system. Some of these curves are plotted in Figure 12. Although they all have the same general shape, they all have different ranges and vary in how rapidly they transition from low to high.

Table 10. Compressive Nonlinearities

CN	Equation
1 [7]	$y = \frac{.05}{1 + e^{-0.521x + 0.613}}$
2 [9]	$y = sgn(x) * 2.75 * \log_{10}(1 + 970\|x\|^{0.69})$
3 [9]	$y = \frac{1}{1-s} * [\frac{1}{1 + e^{\frac{-(x-7.6)}{12}}(1 + e^{\frac{-(x-5)}{5}}} - s]$
	$s = \frac{1}{1 + e^{\frac{7.6}{12}}(1 + e^{\frac{5}{5}})}$
4 [5]	$y = sgn(x) * \min(8000 * \|x\|, 0.06 * \|x\|^{0.25})$
5 [2]	$y = \frac{1}{2} * \sqrt{4 * \|x\|} * sgn(x)$
6 [1]	$y = \begin{cases} 1 + 10 * tan^{-1}(65 * x), x > 0 \\ e^{10*65*x}, x \leq 0 \end{cases}$
7 [10]	$y = \|x\|^{.25} * sgn(x)$
8 [10]	$y = \|x\|^{.75} * sgn(x)$
9 [5]	$y = sgn(x) * \min(18000 * \|x\|, 7.8 * 10^{-3} * \|x\|^{0.16})$

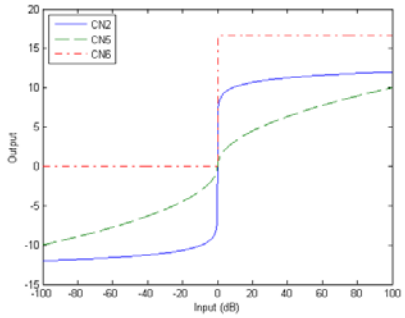

Figure 12. Plots of CN2, CN5 and CN6

As was done for the Baseline Range Function, the compressive nonlinearities are used after log compression. They were implemented as:

$$y = f(x - c) \tag{8}$$

where x represents the input which is the spectral magnitudes of a frame, f represents the compressive nonlinearity, and y is the output. Again, c is the reference point (defined separately for each utterance, as described above), which effectively determines the center point of the nonlinearities.

Sigmoid Function

Another approach to implement compressive nonlinearities is to use a parametrically defined function, with a form and parameters that can make the function effectively be a smoothed version of the BRF or any of the auditory model nonlinearities, by varying parameters in the function. In this work, a sigmoid function with three parameters was chosen, as given in equation 9. In this equation, 100 is an overall gain term (and thus not really important), a determines the steepness of the curve (and effectively the dynamic range), and c determines the center point of the range, with respect to a reference point in the speech. Both a and c can be empirically adjusted, using ASR performance as a metric; however, presumably best values of a and c will correspond to curves similar to some of the auditory model based curves. Based on this thinking, three curves are shown in Figure 5, with value of a = (-.025, -.05, -.075) and c = 0 for this plot. Figures 14 displays spectrograms of 10dB SNR speech after spectral compression with the sigmoid function with a = -0.05, and computed using a t value of 20dB. Figure 15 depicts its effects on a frame of speech. The values for the frame of speech operated on with the sigmoid function were shifted up and scaled so they could be easily compared to the control.

$$y = 100 * \left(\frac{1}{1 + e^{a*(x-c)+a}} \right) \quad (9)$$

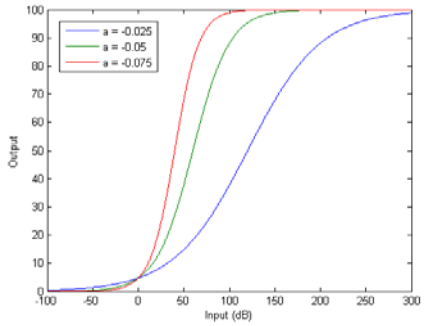

Figure 13. Sigmoid Function

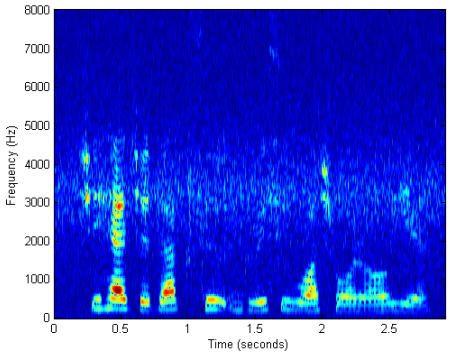

Figure 14. Spectrogram after applying Sigmoid Function (*a* = -0.05)

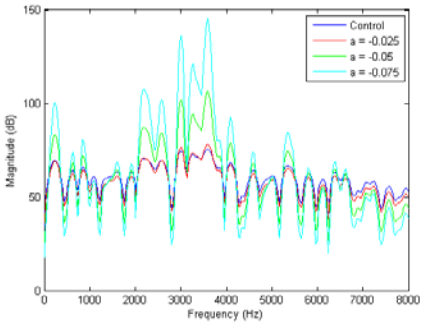

Figure 15. Sigmoid functions of different value of *a* applied on a frame.

3.3.3. Experimental Evaluations

Experiments were conducted to evaluate the various magnitude limiting functions using the TIMIT database. The SA sentences were removed from the database, resulting in 3696 sentences from 462 speakers for training and 1344 sentences from 168 speakers for test. All analysis

parameters were identical for various range limiting functions except for the use of the function themselves.

Left-to-right 3-state HMMs with no skip were used and a total of 48 (eventually reduced to 39 phones) context independent monophones, combining several similar phones as was done in several other works with TIMIT (for example [13]). The HMMs used 16 Gaussian mixtures to model the phonemes. HMMs were created from the training data using the Hidden Markov Model Toolkit (HTK) version 3.4.

The features used are DCTC/DCSC terms from [15]. A total of 39 terms were used, comprised of 13 DCTC terms, each expanded by 3 DCSC terms.

The objective of the experiments was to compare the functions using the phoneme recognition accuracy as a figure of merit for comparison. All accuracies are given as percentages. Testing was done for two general conditions—mismatched (clean training data, varying SNR for test data) and matched (varying SNR for both training and test data.) The controls for both cases are ASR systems with no additional compressive nonlinearity after log compression (Table 11). Table 12 gives results for three of the nonlinearities, after tuning to give best results for mismatched conditions at 10 dB SNR. Results are given, however, for both the mismatched conditions and the matched conditions. Similarly, Table 13 gives results for three of the nonlinearities, but after tuning to give best results for the matched conditions (at 10dB SNR). Again, results, however, are given for both the matched and mismatched cases. The left most column indicates whether the results are matched or not (N is mismatched and Y is matched).

The control results for the mismatched case show that ASR with no additional limiting of spectral magnitudes is extremely sensitive to noise. Even with a small amount of additive noise (30 dB SNR) there is a significant drop in accuracy and at 20 dB SNR, the accuracy is only about half that obtained for clean test data. The additional compressive nonlinearities for spectral amplitudes greatly reduce this drop. However, when the training and test conditions match, even the best compressive nonlinearities are not quite as effective as using the log only.

Table 11. Phoneme Recognition Accuracies for Control Cases

Function	**SNR (dB)**				
	Clean	**30**	**20**	**10**	**0**
Mismatched Control	68.3	54.7	35.0	18.8	6.4
Matched Control	68.3	67.0	63.4	55.8	42.8

3.3.4. Discussion

Nonlinear compression and/or range limiting the log magnitude of the speech spectrum can greatly increase the noise robustness of an ASR system. That is, compression reduces ASR accuracy for systems trained on clean speech and tested on noisy speech. However, this compression step does not appear to be beneficial if the ASR system has matched conditions with respect to noise. The benefits of nonlinear compression of the log spectrum come at a cost of decreased accuracy for matched training and test conditions. If the nonlinearity is tuned for the matched condition, there is still some benefit for the mismatched condition, but not as much as if the nonlinearity is tuned for the mismatched condition.

Table 12. Phoneme accuracies with nonlinearities optimized for mismatched conditions

Match	Function	*t* (dB)	SNR(dB)				
			Clean	30	20	10	0
N	BRF	20	63.5	62.0	53.7	36	16.3
N	CN2	10	57.9	56.1	50.6	32.4	21.3
N	Sigmoid (a=-0.75)	20	61.8	59.4	51.2	34.8	18.6
Y	BRF	20	63.5	62.4	60.4	54.7	42.8
Y	CN2	10	57.9	57.2	55.4	52.0	42.7
Y	Sigmoid (a=-0.75)	20	61.8	60.5	58.0	52.2	42.0

Table 13. Phoneme accuracies with nonlinearities optimized for matched conditions

Match	Function	*t* (dB)	SNR(dB)				
			Clean	30	20	10	0
Y	BRF	40	68.3	66.9	63.3	56.1	42.6
Y	CN2	10	57.9	57.2	55.4	52	42.7
Y	Sigmoid (a=-0.75)	30	63.5	63.2	61.1	55.9	43.6
N	BRF	40	68.3	60.5	40.3	21.4	8.5
N	CN2	10	57.9	56.1	50.6	32.4	21.3
N	Sigmoid (a=-0.75)	30	63.5	61.5	47.3	28.4	14.4

3.3.5. Conclusion

Some methods for limiting the range of spectral magnitudes were demonstrated and shown to improve the noise robustness of ASR systems for mismatched training and test data. It is unclear which function is best, but simply limiting the magnitudes (with a floor) is beneficial. However, this should be more thoroughly tested with more realistic noise types and a larger variety of conditions.

3.4. New Approach to the Hidden Markov Model Decoding Paradigm

3.4.1. Introduction

In most current automatic speech recognition systems, the stochastic modeling structure known as a HMM is the primary tool used to create acoustic models for basic speech units [24]. Features extracted from speech signals are processed and used to train a unique HMM for each phone or word in the recognizer's dictionary. Once trained, these models are used with test data to determine the likelihoods for each possible phone or word. There are many different types of speech features which can be used to train an HMM, some of which are the MFCCs and the DCTCs [15]. For most state-of-the-art speech recognition systems, continuous HMMs are used and thus Gaussian mixtures are employed to approximate the probability density functions of the feature emissions [24]. There are many different HMM types, but they all cluster the speech data into hidden "states" [25]. These states represent a part of the phone or word which is emitting the observed features. Thus, the probability of a feature emission is conditioned on the possible states at each time step. The typical spoken word/phone recognition method is to decode each HMM by means of the Forward Algorithm, which is based (primarily) on feature emission probabilities and (secondarily) on state transition probabilities. The HMM with the highest probability after decoding corresponds to the most likely phone or word.

One potential weakness of the standard HMM method is that HMMs are not trained discriminatively. That is, each HMM is trained only using data from one class, and is trained to best "match" that class, not taking into account the possibility that the HMM may not be very distinct from HMMs of other classes.

In contrast to HMMs, Neural Networks (NNs) are trained discriminatively [26-27]. The basic architecture of a NN includes an input layer with a number of input nodes, followed by one or more hidden layers and an output layer. Design parameters include the number of hidden layers, the number of hidden nodes, the type of node nonlinearities, and details of the training process. The most common use of a NN in ASR systems is to use it a preprocessor to create a more discriminate set of features [28-29]. A NN can also be used to estimate posterior probabilities, which are commonly employed in hybrid NN/HMM systems [28-30]. Although the combination of an NN with a HMM is not a new development, this paper introduces a new approach to the decoding paradigm with regards to the NN outputs.

A NN can be used to classify data into specific words/phones and states. In this paper, a method for combining HMMs with NNs is presented with a new type of HMM decoding. Simulation experiments, as well as evaluations with isolated word recognition experiments, show accuracy improvements with this new method. With refinement, this method may provide a boost in performance for the more general case of continuous speech recognition.

3.4.2. Theory

To provide context and notation, a very brief review of the discrete HMM is presented in the next subsection. It is important to note the difference between discrete and continuous HMMs. Both are represented by a number of discrete time steps; however the emissions of a continuous HMM are characterized by continuous probability densities rather than discrete observations with discrete probabilities as in the former model [24]. For the isolated word recognition experiments reported in this paper, continuous HMMs are used to model each word in the recognizer's vocabulary.

Architecture of Discrete HMM

Discrete HMM are described using three types of probabilities, which are organized into matrices. The first is the transitional probability matrix, which describes the likelihood of transitioning from each state of the HMM to each other possible state. In the Ergodic HMM structure; it is possible to transition from the current state to any one of 'N' states, including same state transitions, at any time step [24]. In the Bakis structure, which is most often used to model speech, backward state transitions are not allowed [24]. The transitional probabilities are modeled using the "State Transition Probability Matrix," which is traditionally denoted as the 'A' matrix [25]. Within each state, the discrete HMM emits one of 'M' observables, according to some probability distribution. These probabilities depend on the state and are described using the "State Emission Probability Matrix" or the 'B' matrix. The final set of probabilities used to describe an HMM are the "Initial State Probabilities" typically summarized and denoted by the matrix 'Π.'

In speech recognition, a sequence of features (the emission sequence) is extracted from the input speech data and decoded by a previously trained HMM. The task here is to determine the likelihood of the sequence with respect to each HMM. This is done using the Forward Algorithm. Sometimes, it is also useful to look at the most likely state sequence given an observation sequence in which case the Viterbi Algorithm can be employed [24].

It is important to mention that the typical method of training is the Baum-Welch algorithm. However, because this paper mostly deals with the decoding of HMM the details of training are not discussed. For a more in depth tutorial on HMM training and decoding issues, the reader may see [24]. Although these algorithms work reasonably well, they seek to maximize the joint probability of a given emission sequence and likely state sequences using the probability of an

emission given a state [25]. It seems that a more direct approach would be to maximize the same probability, but instead using the probabilities of state conditioned on emissions, provided such probabilities can be estimated, since the emission sequence is observable.

A different perspective on the discrete HMM

If the posterior probabilities describing the likelihood of being in each possible state given an emission can be found, they would appear to be useful for decoding. For the discrete HMM case they can be stored in matrix form as the "State Conditional Probability Matrix," which we denote as the 'C' matrix. A method for decoding an HMM given these probabilities can be derived beginning with the chain rule of probability. Two assumptions are required. The first is the first order Markov assumption, which is that the transition probabilities depend only on the previous state and not the states before that [25]. The second is the conditional independency of observable parameters which is that the probability of an emission is dependent only upon the current state of the model and not the previous states or emissions. Both of these assumptions are also made during normal HMM decoding and the only change here is that they are applied to a different set of probabilities. With q_T representing a state and v_t representing a discrete emission with time sub indices, for a T time step model, the derivation begins by applying the chain rule to the joint probability of a state sequence and emission sequence:

$$P(q_T, \ldots, q_1, v_T, \ldots, v_1) = P(q_T | v_T, \ldots, v_1, q_{T-1}, q_1) P(q_{T-1}, \ldots, q_1, v_T, \ldots, v_1) \quad (10)$$

By applying the aforementioned assumptions, the equation can be rewritten as:

$$P(q_T, \ldots, q_1, v_T, \ldots, v_1) = P(q_T | v_T) P(q_{T-1}, \ldots, q_1, v_{T-1}, \ldots, v_1) \quad (11)$$

The chain rule is applied repetitively, using the assumptions, and the equation becomes the simplified version:

$$P(q_T, \ldots, q_1, v_T, \ldots, v_1) = P(q_T | v_T) P(q_{T-1} | v_{T-1}) \ldots P(q_1 | v_1) P(v_1) \quad (12)$$

In typical HMM decoding, it is assumed that the model begins in state 1, thus the marginal probability '$P(v_1)$' will affect all possible paths equally. This means that it will not factor in the

decision of the most probable state at this or any subsequent time step and therefore can be removed from the equation. Another method of interpreting this probability is outlined in the following section. Assuming that the probability $P(v_1)$ is unnecessary, equation 12 becomes:

$$P(q_T, \ldots, q_1, v_T, \ldots, v_1) = P(q_T|v_T)P(q_{T-1}|v_{T-1}) \ldots P(q_1|v_1) \quad (13)$$

A key difference between decoding based on equation 6 and the standard decoding method is the lack of transition probability terms. Since the transition matrix and the conditional matrix represent the probability of being in a certain state, conditioned on different variables, and different assumptions, they should not be combined. However, in practice when the Bakis structure is assumed, there must be some way to disallow "backward" state transitions. To accomplish this, a binary matrix of ones and zeros, where zeros represent a forbidden state transition and ones an allowable, can be employed to prevent the algorithm from choosing an impossible path. This matrix has no effect on the probability calculations other than to remove impossible paths through the HMM from consideration. This type of transitional matrix, dubbed "State Path Disabling Matrix," is denoted as 'D.'

The decoding algorithm using the C and D matrices was named the ViterbiC algorithm. There are two variations of this algorithm which are analogous to the forward algorithm and the Viterbi algorithm (model likelihood and probable path respectively). Since the simulations were done using the path identification algorithm, that is the version which is described below. It can be divided into four steps; however, the final two steps ('Termination' and 'Path Backtracking') are the same as in the standard Viterbi Algorithm and are not included:

$$\textit{Initialization:} \quad \delta_1(i) = c_1(v_1) \quad (14)$$

$$\Psi_1(i) = 0$$

$$\textit{Propagation:} \quad \delta_{t+1}(j) = \max_{1 \le i \le N} \delta_t(i) c_j(v_t) d_{ij} \quad (15)$$

$$\Psi_{t+1}(j) = argmax_{1 \le i \le N} \delta_t(i) c_j(v_t) d_{ij}$$

In the case of the Ergodic structure, the D matrix becomes a matrix of all ones and thus can be removed from the equations. However, when certain state transitions are forbidden the D matrix must remain. One concern was how to define the $P(v_1)$ component in equation (12) and whether it was necessary. Further investigation into this problem led to the creation of a second

algorithm which uses a combination of the Viterbi and ViterbiC algorithms and was dubbed the "Hybrid Algorithm."

The hybrid algorithm

Given that the emission, transition, prior and conditional probabilities are known (or can be estimated), a method which utilizes all the information when it determines the most probable path would appear to be most promising. The Hybrid Algorithm was developed in an attempt to use the $P(v_1)$ term in equation (12). At major difference between the ViterbiC algorithm and the Hybrid algorithm is that at each time step, the probability of the observed symbol $P(v_t)$ is calculated and factored into the total likelihood, but is state dependent. To calculate this probability; the definition of conditional probability and Bayes theorem are applied, which results in:

$$P_j(v_t) = \frac{P(q_j) * P(v_t|q_j)}{\sum_{j=1}^{N} P(q_j) * P(v_t|q_j)} \tag{16}$$

The most probable path is then calculated using a four step method, however only the first two steps are shown below. The final steps 'Termination' and 'Path Backtracking' are performed in the same manner as for the Viterbi Algorithm:

Initialization:

$$P_i(v_t) = \frac{\pi_i b_i(v_1)}{\sum_{i=1}^{N} \pi_i b_i(v_1)} \tag{17}$$

$$\delta_1(i) = P_i(v_t) * c_i(v_t)$$

$$\Psi_1(i) = 0$$

Recursion:

$$\Delta_t(j) = \max_{1 \le i \le N} \delta_{t-1}(i) * a_{ij} \tag{18}$$

$$\Psi_t(j) = argmax_{1 \le i \le N} \quad \delta_{t-1}(i) * a_{ij}$$

$$P_j(v_t) = \frac{\Delta_t(j)b_j(v_t)}{\sum_{j=1}^{N} \Delta_t(j)b_j(v_t)}$$

$$\delta_t(j) = P_j(v_t) * c_j(v_t)$$

Although the hybrid algorithm requires more computations than the ViterbiC, it does utilize all the information that is known about the HMM. It is worth noting that the transitional properties of the 'D' matrix from the ViterbiC algorithm are captured in the state transition matrix 'A' and thus the inclusion of the 'D' matrix is unnecessary.

3.4.3. Monte Carlo Simulation

A Monte Carlo simulation was designed to compare the performances of the Hybrid, ViterbiC and the Viterbi algorithm (along with several other variations of the first two). The purpose was to determine whether these new decoding methods were effective compared to the standard Viterbi decoding using random emissions created by a specified HMM. The simulations were designed using the MATLAB HMM toolbox and performed for the Ergodic discrete HMM structure.

For each iteration, the "true" transition and emission matrices were randomly generated (with the emission probabilities sampling a Gaussian distribution). Each model was composed of 5 states and 50 observables. Training data of 10^8 time steps was generated and the transition/emission matrices were re-estimated using the Baum-Welch algorithm. The conditional probabilities were estimated by summing the number of times an observable appeared with a given state divided by the number of times that observable appeared in total for the training data. Once the matrices were estimated, one hundred Markov chains were generated using the true transition/emission matrices. Each chain was decoded using every algorithm (with its appropriate estimated matrices) with the intent of finding the most accurate state transition path. The algorithm with the most accurate path for each Markov Chain received a "point." At the end of one round (one hundred decoded Markov chains), the algorithm with the highest number of points received a "win." The method was repeated for fifty iterations and the number of "wins" was recorded.

The experiment was performed three times and the noteworthy results are shown in Table 14. As a side note, several other versions of these algorithms were also evaluated; however none of these other algorithms performed as well as those for which results are given. Each algorithm was scored individually and thus the maximum value possible for each score was fifty.

Table 14. Monte Carlo Simulation Results—Number of "wins"

Viterbi	ViterbiC	Hybrid
0	11	30
2	11	28
0	15	29

In these simulations, the Hybrid algorithm clearly performed the best, with the ViterbiC scoring a respectable second place. Both algorithms performed significantly better than the standard Viterbi Algorithm. The results from the simulations showed that, for the conditions tested, the new decoding algorithms result in more Markov chains decoded accurately than are obtained with the standard method.

3.4.4. Experiments with Speech Data

Since the results of the Monte Carlo simulations were promising, an experiment designed to test this method with Isolated Word Recognition (IWR) using speech data was devised. The pilot database was simply one speaker and the digits 0-9. For every word, 65 DCTC/DCS features were computed [15]. To compare the performances of the new decoding algorithms with the standard HMM method a Bakis structured, continuous HMM was trained using the Murphy HMM open source toolkit [31]. Each word was assigned five states and constrained to begin in state one. The PDF of the emission densities were approximated using nine Gaussian mixtures as suggested in [24]. Using the Forward Algorithm, the HMM was able to classify 100% of the training data correctly and 98% of the testing.

The key difference between the speech data and the previously-described simulations was the method of estimating the posterior probabilities for the conditional matrix. In this experiment, a neural network was configured to classify the data by word and state. Since the inputs to the NN were the features, the outputs of the NN could be viewed as a classification of word and state, given the feature values (emissions). These outputs approximate the conditional probabilities which were used in the simulations.

With a NN used to estimate the posterior probabilities, four decoding algorithms were used to classify the testing data. The first was a forward ViterbiC algorithm which simply summed the possible paths at each node. The second was a most probable path seeking ViterbiC algorithm which found the most likely path and used that as a representative for each word. The word classification was then performed by comparing each word's representative path probability and the most likely path was then the chosen word. The third algorithm was a Hybrid forward algorithm and the fourth was a path seeking Hybrid algorithm.

The neural network architecture

A key requirement for the decoding method introduced in this paper is that the posterior probabilities for the 'C' matrix be well estimated. For the given training data, it was found that a NN with five hidden layers and fifty five hidden nodes per layer resulted in the most accurate classification of the training and testing data (99.6% and 78% respectively). The nodal output function at each layer was the 'tansig' function. The inputs to the NN were 65 DCTC/DCS features and the output was configured to be a fifty row matrix representing the five states of each of the ten words. To train the NN, state boundaries were used to create a target matrix, which were defined as the most likely state transition paths for all the training data (found using the Viterbi algorithm and "normally" decoded HMMs).

3.4.5. Results

With regards to the pilot database, in order for the new decoding method to be considered successful, at least one algorithm should perform better than the standard HMM decoding on the testing data (recall this performance was 98%). The results for this experiment are shown below in Table 15.

Table 15. Pilot Database IWR Results

	Forward ViterbiC	ViterbiC Path	Forward Hybrid	Hybrid Path
Training Data:	100%	100%	100%	100%
Testing Data:	99%	100%	98%	99%

The results for the pilot database were very positive; showing that with reasonably accurate neural network performance, each of the decoding methods matched or beat the standard HMM forward algorithm method. In particular, the ViterbiC path seeking algorithm was able to classify all of the data correctly.

The next step was to make the task more difficult. It was decided to keep the number of words in the database the same, but to increase the number of speakers to thirty speakers per word. The data for this was taken from the OGI Isolet Database [32]. The dictionary consisted of the first ten letters of the alphabet (A-J) with thirty training and testing files per word. One file from each of the thirty speakers in the database was used in training and another in testing. Once again, 65 DCTC/DCS features were computed and an HMM was trained which yielded a performance of 83.3% on the training data and 72% on the testing data, forming a baseline. The architecture of the NN was changed to just one hidden layer with 50 hidden nodes. In addition to the new architecture, the outputs of the NN were transformed using a 'softmax' transfer function as

recommended in [28]. The NN was trained and resulted in a performance of 45.1% on the training data and 40% on the testing data. The outputs were then used in the same four decoding algorithms and the results are shown in Table 16.

Table 16. Expanded Database IWR Results

	Forward ViterbiC	ViterbiC Path	Forward Hybrid	Hybrid Path
Training Data:	86.7%	81%	87.3%	81%
Testing Data:	76.3%	76.7%	71.3%	71.3%

The results for the expanded database are promising. The Hybrid Forward Algorithm beat the standard method of HMM decoding by 4.7% on the testing data. The ViterbiC Forward Algorithm also beat the standard HMM decoding by 4.3%. Both of these algorithms outperformed the standard method, despite the poor performance of the NN to classify both the training and the testing data.

3.4.6. Conclusions

The results from the expanded database IWR indicate a potential performance gain by decoding HMMs in this way. If the accuracy of the NN is improved, it is expected that the results for the IWR will improve as well. Ways to refine the NN are being investigated (such as using "don't cares' for NN targets to account for uncertainties in state boundaries) and will be implemented with the goal of increasing the recognition rates of these algorithms. After these refinements are made, the new decoding methods will be applied to continuous speech and to acoustic modeling at the phone level.

4. RESULTS AND DISCUSSIONS

Each subsection of the main body of the report has results and discussion. In terms of the database development, 300 video clips in each of English, Mandarin, and Russian have been collected and annotated at various resolution levels. The forced alignment procedure has been shown to give accurate results for phonetic level labeling, using comparisons with subsets of data that were manually labeled at the phonetic level. All of the algorithmic methods investigated show promise for improving ASR accuracy and/or robustness.

5. CONCLUSIONS

Each subsection of the main body of the report has its own conclusion.

6. REFERENCES

[1] Zue, V. and Seneff, S., "Speech database development at MIT: TIMIT and beyond," *Speech Comm.*, 9, 351-356, 1990.

[2] Graff, D., "An overview of Broadcast News corpora," *Speech Comm.* , 37(1-2): 15-26, 2002.

[3] Garofolo, J. S., Laprun, C. D. and Fiscus, J. G., "The rich transcription 2004 spring meeting recognition evaluation," *NIST 2004 Spring Rich Transcription Evaluation Workshop*, Montreal, Canada, 2004.

[4] Cohen, J., "The Gale Project: A description and an update," *ASRU IEEE Workshop on Automatic Speech Recognition & Understanding*, 237-237, Dec. 2007.

[5] Paulsson, N., Choukri, K., Mostefa, D., Dipersio, D., Glenn, M. and Strassel, S., "A large Arabic broadcast news speech data collection," *MEDAR 2nd International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 22-23 April, 2009.

[6] http://en.wikipedia.org/wiki/YouTube

[7] Barras, C., Geoffrois, E., Wu., Z and Liberman, M., "Transcriber: Development and use of a tool for assisting speech corpora production," *Speech Comm*., 33(1-2): 5-22, 2001.

[8] http://projects.ldc.upenn.edu/gale/Transcription/

[9] http://mandarin.about.com/od/chineseculture/a/dialects.htm

[10] B.E.D. Kingsbury, N. Morgan, and S. Greenberg, "Robust Speech Recognition using the Modulation Spectrogram," *Speech Comm*. 25, pp.117-132, 1998.

[11] F. Valente, and H. Hermansky, "Hierarchical and Parallel Processing of Modulation Spectrum for ASR Applications," *Proc. ICASSP 2008*, pp.4164-4168, 2008.

[12] H. Hermansky, and N. Morgan, "RASTA Processing of Speech," *IEEE Trans. Speech and Audio Processing*, 2(4), pp.578-589, 1994.

[13] K.F. Lee, and H.W. Hon, "Speaker-independent Phone Recognition using Hidden Markov Models," *IEEE Trans. on Acoustic, Speech and Audio Processing*, 37(11), pp.1641-1648, 1989.

[14] N.Kanedera, and H. Hermansky, "On Properties of Modulation Spectrum for Robust Automatic Speech Recognition," *Proc. ICASSP*, pp.613-616, 1998.

[15] S.A. Zahorian, H. Hu, Z. Chen. and J. Wu, "Spectral and Temporal Modulation Features for Phonetic Recognition," *Interspeech 2009*, pp. 1071-1074, 2009.

[16] Haque, S., Togneri, R. and Zaknich A., "Perceptual features for automatic speech recognition in noisy environments," Speech Communication 51, pp. 58-75, 2009.

[17] Harte, M. J., Elliot, J. S. and Rice, J. H., "A comparison of various nonlinear models of cochlear compression," J. Acoust. Soc. Am., 117 (6), pp. 3777-3786, 2005.

[18] Kim, D., Lee, S. and Kil, M. R., "Auditory Processing of Speech Signals for Robust Speech Recognition in Real-World Noisy Environments," IEEE Trans. Speech and Audio Processing, (7) 1, pp. 55-69, 1999.

[19] Tchorz, J. and Kollmeier, B., "A model of auditory perception as front end for automatic speech recognition," J.Acoust. Soc. Am., 106 (4), pp. 2040-2050, 1999.

[20] Sumner, J. C., Lopez-Poveda, A. E., O'Mard, P. L. and Meddis, R., "A revised model of the inner-hair cell and auditory nerve complex," J. Acoust. Soc. Am., 111 (5), pp. 2178-2188, 2002.

[21] Dau, T., Püschel, D. and Kohlrausch, A., "A quantitative model of the "effective" signal processing in the auditory system. I. Model structure," J. Acoust. Soc. Am., 99 (6), pp. 3615-3622, 1996.

[22] Chiu, B. Y., Bhiksha, R. and Stern, M. R., "Towards Fusion of Feature Extraction and Acoustic Model Training: A Top Down Process for Speech Recognition," Interspeech, pp. 32-35, 2009.

[23] Chiu, B. Y. and Stern, M. R., "Analysis of Physiologically-Motivated Signal Processing for Robust Speech Recognition," Interspeech, pp 1000-1003, 2008.

[24] Rabiner, Dr. Lawrence R. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." Proceedings of the IEEE, vol. 77, no. 2, February 1989

[25] Cho, Dr. Sung-Jung. "Introduction to Hidden Markov Model and its Application." Lecture Slides, 16 April 2005. Samsung Advanced Institute of Technology (SAIT).

[26] Lippman, Dr. Richard P. "An Introduction to Computing with Neural Nets." IEEE ASSP Magazine, April 1987.

[27] Hush, Don R. et al. "Progress in Supervised Neural Networks: What's New Since Lippmann." IEEE Signal Processing Magazine, January 1993.

[28] Hermansky, Hynek. Et al. "Tandem Connectionist Feature Extraction for conventional HMM Systems." Oregon Institute of Science and Technology, Portland, Oregon, USA. International Computer Science Institute, Berkeley, California, USA. ICASSP-2000, Istanbul.

[29] Bengio, Yoshua. Et al. "Global Optimization of a Neural Network-Hidden Markov Model Hybrid." IEEE Transactions on Neural Networks, Vol. 3, No. 2, March 1992.

[30] Singer, Elliot. Richard P. Lippman. "A Speech Recognizer Using Radial Basis Function Neural Networks in an HMM Framework." Lincoln Laboratory, MIT. Lexington, MA 02173-9108, USA

[31] Murphy, Dr. Kevin. Hidden Markov Model (HMM) Toolbox for MATLAB. MIT. Updated 8 June 2005. "http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html"

[32] OGI ISOLET (Isolated Letter Speech Recognition). Cole, Ron. Fanty, Mark. Dept. of Computer Science and Engineering. Oregon Graduate Institute, Beaverton, OR 97006, September 12, 1994

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

A1 S. A. Zahorian, J. Wu, M. Karnjanadecha, C. Sekhar Vootkuri, B. Wong, A. Hwang, E. Tokhtamyshev, "**Open Source Multi-Language Audio Database for Spoken Language Processing Applications**," *Interspeech* 2011, August, 2011

Open Source Multi-Language Audio Database for Spoken Language Processing Applications

Stephen A. Zahorian, Jiang Wu, Montri Karnjanadecha
Chandra SekharVootkuri, Brian Wong, Andrew Hwang, Eldar Tokhtamyshev
Department of Electrical and Computer Engineering, Binghamton University, USA
{zahorian,jwu1,kmontri,cvootku1,bwong5,ahwang1,etokhta1}@binghamton.edu

Abstract

Over the past few decades, research in automatic speech recognition and automatic speaker recognition has been greatly facilitated by the sharing of large annotated speech databases such as those distributed by the Linguistic Data Consortium (LDC). Open sources, particularly web sites such as YouTube, contain vast and varied speech recordings in a variety of languages. However, these "open sources" for speech data are largely untapped as resources for speech research. In this paper, a project to collect, organize, and annotate a large group of this speech data is described. The data consists of approximately 30 hours of speech in each of three languages, English, Mandarin Chinese, and Russian. Each of 900 recordings has been orthographically transcribed at the sentence/phrase level by human listeners. Some of the issues related to working with this low quality, varied, noisy speech data in three languages are described.

Index Terms: open source speech database; forced alignment; transcribe; speech recognition

1. Introduction

The need for large well-labeled databases for spoken language processing is well known. "There is no data like more data." is a comment made by MIT speech researcher Victor Zue at a speech recognition workshop in the 1980s. Despite the large number and vast sizes of speech databases developed since the 1980s, the comment by Victor Zue still rings true [1].

One of the first large databases developed for speech research was the TIMIT acoustic-phonetic continuous speech corpus. With joint efforts from Texas Instruments, SR International and MIT, TIMIT was published by the LDC in 1993. This database contains recordings of 630 speakers, each reading 10 sentences, in "studio" conditions. The data was then manually labeled with starting and ending points for each phone in each sentence. Even today, TIMIT is one of the most widely used speech corpora for phonetic level speech research.

However, the 5,040 sentences in TIMIT (the SA sentences are often removed), with a typical sentence duration of 5 seconds, only provide about 7 hours of total speech, which is insufficient for many recognition tasks. Since TIMIT, many other speech databases have been collected, transcribed, catalogued, and distributed by the LDC. For example, the English Broadcast News Transcripts (HUB4) [2] database was launched by the LDC in 1996 and now contains approximately 100 hours of broadcast news and has all speech manually transcribed at the phrasal level, using the Rich Transcription (RT-04S) Evaluation Data guidelines developed by the National Institute of Standard and Technology (NIST) in 2004 [3]. In recent years, LDC announced their plan for developing a new speech database. The DARPA GALE program [4] is a very large speech database project with the goal of collecting speech in multiple languages from global broadcast news. In 2009 the LDC [5] reported that 4,000 hours of Arabic broadcast has been collected and 2,400 hours were selected for transcribing.

Currently, public video sharing websites such as YouTube are booming because sharing homemade videos has become very easy and more popular. About 65,000 videos have been uploaded daily since 2006, and this number continuously increases [6]. This seemingly "infinite" number of videos found on the web can provide a vast collection of speech data for speech research, and the topics and speaking styles corresponding to these collections are much more varied than those found in broadcast news. To tap into this large resource, an "open source multi-language speech database" project was developed and is described in this paper.

2. Structure of the database

2.1. The goals

The database was developed with collections from three different languages: English, Mandarin Chinese, and Russian. The intent was to collect about 30 hours of speech in each language, consisting of 300 videos per language, with videos averaging about 7 minutes in duration. The intent was also to collect three videos from each of 100 speakers per language, with the three recordings from each speaker originally spoken on different days and under different recording conditions. Another goal was to have approximately an equal number of male and female speakers. As is discussed later, most but not all, of these guidelines were met. The only firm guidelines were that the audio portion of each video be of sufficient quality to be "reasonably" intelligible by a "typical" native speaker of the language, that there not be constant background noise (i.e., not have background music throughout the entire passage), and that no single passage be shorter than 1.5 minutes or longer than 16 minutes in duration. Since many videos were in fact longer than 16 minutes, a "stand-alone" primarily speech portion of the video was extracted (using Xilisoft Video converter) for the database.

2.2. Video download and post-preparation

The first step in this development was to identify, download, and store audio/video clips from public video sharing websites. All videos were downloaded in the highest quality format that the sharing sites supported and then stored in a standard format. Table 1 shows the typical sites used for each language, the download tools used, and the original file formats.

For those videos in a format other than MP4, further processing was done, so that all videos were saved as MP4 files. The step was done by Xilisoft video converter, too. A copy of each video in its original format was also kept. Then all files were designated with a specific name based on the language, the gender of the speaker, the initials of the speaker, and the category of recording. The category of recording

comprised "formal presentation" and "casual conversation" which refer to the conditions under which the recording was made. Table 2 shows some criteria for making a decision on which category a video clip belonged to, although the decisions were somewhat subjective.

Table 1. *Typical video sharing websites and tools for downloading.*

Language	*Typical Webs*	*Download Tools*	*Original Format*
English	Youtube.com	www.savevid.com	MP4
Chinese	Youku.com	www.flvcd.com	FLV
Russian	Rutube.ru	www.savevid.com	MP4

Table 2. *Some criteria for separating "Formal Presentation" and "Casual Conversation"*

	Formal Presentation	***Casual Conversation***
Speech type	Speech prepared in advance	Casual talk, semi-spontaneous
Noise/Interruptions in recording environment	Quiet and not much noise	More noise and even distortion effects
Slang/Disfluencies	Very little	Usually a lot
Background music	None	Maybe a little

A standard file name consists of 5 parts, chosen to make it easier to sort and organize the videos. Figure 1 illustrates a file name given to a Chinese language video sample, which is categorized as a "casual conversation," spoken by a male speaker who has initials "WX." The detailed description and possible options for each part as well as their abbreviations in file name are given in Table 3.

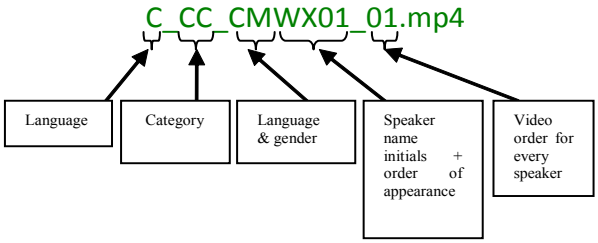

Figure 1. *Illustration of naming conventions for video files*

Table 3. *Filename notation and descriptions*

	Description	***In filename***
Language	English	E
	Mandarin Chinese	C
	Russian	R
Recording Category	Formal Presentation	FP
	Casual Conversation	CC
Gender	English/Chinese/Russian	E/C/R
	Male/Female	M/F
Speaker Name Initials	Initials of first / last name	
	Order of appearance in database	01, 02, 03…
Video Order	Indicates which video of each speaker	01, 02, 03…

3. Transcribing the database

An important aspect of this database is that all speech files were manually transcribed by human listeners. The reason for this was quite simple: given the relatively low quality, the wide variations in recording conditions, the presence of background noises, and the multiple languages, it seemed very doubtful that any automatic speech recognizer would be able to establish reliable "ground truth." By carefully listening to each sentence and reviewing by different listeners, the human transcriptions would provide the best orthographic transcriptions of this "ground truth." Also, the accurately transcribed sentences provides the best starting point for an automatic forced alignment process to create time labels for words and phonemes, thus better supporting phonetic recognition research. Therefore, properly selecting the tool and building the specifications for the transcribing work was necessary.

3.1. Transcribing tool

Transcriber, developed as a tool for assisting in creating the speech corpora in [7], was designed for manual segmentation and transcription of long duration broadcast news recordings, including annotation of speech turns, topics and acoustic conditions. With its embedded user-friendly graphical user interface, Transcriber allows listeners to perform tedious and complex operations such as modifying the time boundary of each speech segment, adding noise notations or indicate switching between speakers in a convenient way. Also, the output of Transcriber accurately records the time durations between segments, which provide support for the following automatic forced alignment process for detailed word and phonetic transcription. All these features made Transcriber extremely well-suited for the transcription task.

Other speech transcription tools considered are listed in Table 4. Although Transcriber does not have all the functions these other tools have, it does have the required features and there is no licensing fee; therefore, Transcriber 1.5.1 was used for this project.

Table 4. *Other available transcription tools*

Tool	***Main Features***
XTrans	Multi-speakers tasks (Developed by LDC)
Transana	Link the transcription place to video
SoundIndex	Directly transcribe in XML file
WaveSurfer	Waveform display/analysis

3.2. Audio preparation

Transcriber only reads WAV files as its audio input. Therefore audio-only WAV files were extracted from the video MP4 files using a software tool called AOA audio extractor. Factors contributing to overall speech quality and intelligibility include: the background noise and recording conditions when videos were made, the loss by website compression tools for uploading files from users, another round of compression by the video download tool, and the final audio extraction. In order that the only significant degradation be due to the first two factors, high quality settings were used for the last two steps. Tables 5 and 6 list the quality setting for downloads and the quality setting for audio extraction, respectively.

Table 5. *Video quality specifications*

Format	***MPEG-4***	
Video	Encoding	MPEG-4 AVC1 (H.264)
	Resolution	Original as on YouTube
Audio	Encoding	AAC
	Channels	2
	Sampling rate	44100Hz

Table 6. *Audio quality specifications*

Format	***WAV***	
Audio	Encoding	PCM
	Channels	2
	Sampling rate	22050 Hz
	Bits/sample	16

3.3. Metadata and annotation

The annotation and metadata for transcribing this database was based on the format used for the LDC project GALE [8]. Unlike studio quality recorded read speech, there is a large amount of variability in transcribing web-collected speech. Main factors which possibly diminish the transcribing quality included the background noise/music, slang, and inserted words in different languages, other than the primary language of the speaker. These additional factors were annotated to the extent feasible.

When selecting videos, background music was considered permissible if it was not "too" high level and did not overlap with speech too often. Music as well as other types of noise were labeled in the transcription using the notation in Table 7. Noise labels differed depending on when the noise occurred relative to the speech. "Burst" noise/music occurred when there was no speech. "Overlap" noise/music occurs during speech. The most commonly used notation for noise was the "Others" notation, since, from listening, it was often quite difficult to accurately determine the source or type of the noise. Many speech signals also contained static noise which was present throughout the signal. This was labeled in a master file that describes each file.

Table 7. *Notations for noise and how it is annotated*

Noise Notation	*Description/Example*	*Symbol/burst*	*Symbol/overlap*
Music		*[mu]*	*[mu-][-mu]*
Applause		*[a]*	*[a-][-a]*
Laughing		*[l]*	*[l-][-l]*
Human	coughing, inhaling,	*[h]*	*[h-] [-h]*
Nature	wind, ocean...	*[n]*	*[n-] [-n]*
Vehicle	honks, car engine…	*[v]*	*[v-][-v]*
Animal	barking, birds…	*[an]*	*[an-] [-an]*
Office	telephone…	*[o]*	*[o-] [-o]*
Machinery	fan, construction...	*[m]*	*[m-] [-m]*
Others		*[ot]*	*[ot-] [-ot]*

Additionally, special events like silence segments and language transitions are potentially as detrimental as noise or other interference. While transcribing, the listeners labeled a "pause" (Table 8) as a speech break (silence) between 0.5 and 1 second. Silences longer than 1 second were labeled differently. For some run-on sentences which commonly occur in casual conversation, great care was taken with the time labels because speech often does not end abruptly, but rather gradually fades.

Table 8. *Notations for important events and how it is annotated*

Event	*Description/Example*	*Symbol*
Pause	Break btw/ .5 to 1 sec	*[p]*
Language Transition	Word(s) spoken in different language	*Exp. [lang=English-] [-lang=English]*

One common issue in spoken Mandarin Chinese is that people mix English words in Chinese sentences. (In constrast, Russian speakers often use English word variants.) Thus, labeling of languages transitions became necessary. Table 8 also shows the symbol for a language transition.

The issues related to slang, truncated words, incomplete pronunciation and other informality in spoken language are clearly addressed by the GALE standard. These issues include: contractions or ambiguous words should be transcribed as closely as possible to how they actually sound; truncated words are to be ended with a dash "-"; the notation "(())" is to be used to represent an unintelligible words; and tilde "~" is to be used when each letter of an acronym is spoken individually

4. Practical issues in transcribing

4.1. Common issues

The most difficult issue in the transcribing process of all three languages stemmed from the great range of qualities in the original video recordings. For example, some videos downloaded from YouTube had High-Definition quality, which has very high resolution for both audio and video. However, most videos were recorded with much lower quality for both video and audio. Some Chinese videos were recorded with defective equipment, resulting in distortion of the speech signal thus causing difficulties even for a human listener.

4.2. English

In English, speakers often fill pauses with fillers such as "um" or "hm," briefly between words, or extensively to gain time for a next thought. Differences in pronunciation due to culture and accent promoted its own share of concerns; various accepted norms of speech (i.e. tomato), and the use of foreign language for brand names, proper nouns, descriptions, and verbs are used in conjunction with English speech. Furthermore, background noise or feedback from the medium used to record video was an additional factor.

Simplifying words and expressions through slang was very common; often times words that end in "-ing" are mispronounced and are transcribed as "-in." For example "sleepin" for "sleeping." And the use of "dope" and "hot" to express emotions or quality. The widespread use of internet acronyms such as "brb" and "lol" occurred occasionally in causal speech, implying the assimilation of today's digital jargon in verbal communication.

4.3. Mandarin Chinese

Other than embedded English words, the transcription of Mandarin Chinese was done using standard Chinese characters for transcriptions. Unlike English, there are no slang or informal language (such as truncated words) related ambiguities in the transcriptions.

However, the Chinese language has a large number of dialects with a resulting big influence on how people pronounce Mandarin [9]. Accents among Mandarin speakers differ significantly. For example, the speakers from the northeast region of China confuse "s-" [s] and "sh-" [ʂ], and also there was no clear boundary between the nasal consonants "-ng" [ŋ] and "-n" [n] for south China speakers. In the development of the database described in this paper, the listeners always transcribed according to actual pronunciation, even if matching with its neighbor characters did not make linguistic sense.

4.1. Russian

In daily life, some words of the Russian language have a different pronunciation than defined in the dictionary. For example, “тыща” ['tɨɕɕa] is often pronounced “тысяча” ['tɨsjtɕa], “щас” [ɕɕas] as “сейчас” [sej'tɕas], and “сёдня” ['sjdnja] as “сегодня” [se'gondnja]. Similar to Chinese and English transcriptions, all these ambiguous words were transcribed as pronounced.

Russians use English words directly sometimes. If the English word is clear with the correct pronunciation, the words are enclosed with English language tags. However, in some cases English words are misused (“Americanisms”): in such cases words are transcribed as they sound in Russian. Also, the Russian language includes some words with a Ukrainian pronunciation. Such words were marked with double parentheses.

5. Summary table of data collected

The goal of 300 videos for each language had been achieved, and roughly 30 total hours of speech for each database has been collected and transcribed, as summarized in Table 9. In the past 9 months, which included about 3 weeks for data collection and approximately 8 months for transcribing, overall, 11 students were involved in the database development.

Table 9. *Summary of Database*

Language	*Speech Type*	*Gender*	*Total Num. of speakers*	*Total Num. of Recordings*	*Total Time*
English	Formal	Male	88	124	14 hrs
	Formal	Female	21	26	2.6hrs
	Casual	Male	36	108	10.5 hrs
	Casual	Female	14	42	4 hrs
		Total	**159**	**300**	**31.1hrs**
Chinese	Formal	Male	59	136	11.1hrs
	Formal	Female	24	56	4.4 hrs
	Casual	Male	44	82	6.4 hrs
	Casual	Female	10	26	2 hrs
		Total	**137**	**300**	**23.9hrs**
Russian	Formal	Male	79	167	20.5hrs
	Formal	Female	26	42	4hrs
	Casual	Male	36	71	8.1hrs
	Casual	Female	17	20	2.2hrs
		Total	**158**	**300**	**34.8hrs**

6. Use of Forced Alignment for Speech Labeling

The goal of this database collection project is to make the database useful for speech processing research. In order to fulfill that goal, the database must provide accurate word-level and phone-level transcriptions, both with time marks. Although the speech data could be transcribed manually by a well-trained phonetician, the manual method is a laborious and time-consuming task, which makes it impractical for a large amount of data. An efficient way of achieving this is to apply automatically derived forced alignment for this more detailed labeling and then use humans to finalize the labeling.

By using an Automatic Speech Recognizer (ASR) “tuned” for a single passage or group of passages, that is, given phonetic models and word models in terms of phonetic lattices, the audio can be accurately time aligned with word transcriptions. The recognizer can be configured to give a best time-aligned match between the audio and transcription using all the probabilistic constraints imposed by the phonetic and word models. Furthermore, this recognizer can be made much more accurate by configuring it for each recording. For example, based on the human transcription of a recording, the vocabulary can be restricted to only the words in that recording, and the language model derived only from the word and word-pair frequencies in that passage. We are currently experimenting with several techniques for forced alignment on a subset of the English database and the outcome looks very promising. These techniques are being tuned for this particular task and database.

7. Conclusion

In this research project, a large database of English, Mandarin, and Russian was collected, formatted, organized, annotated, and given time-aligned orthographic transcriptions at the sentence/phrase level. Due to the variability, noisiness, and low speech quality, human listeners were employed for this transcription and annotation. Automatic speech recognition techniques will be developed and used to aid in the annotation process at a more fine-grained level. This database will be useful for both automatic speech recognition research and automatic speaker recognition research. The database is derived from open source public web sites; thus it is a sampling of an “infinite,” widely accessed repository of speech.

8. Acknowledgements

This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-10-2-0160. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

9. Disclaimer

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

10. References

[1] Zue, V. and Seneff, S., “Speech database development at MIT: TIMIT and beyond,” *Speech Comm.*, 9, 351-356, 1990.

[2] Graff, D., “An overview of Broadcast News corpora,” *Speech Comm.* , 37(1-2): 15-26, 2002.

[3] Garofolo, J. S., Laprun, C. D. and Fiscus, J. G., “The rich transcription 2004 spring meeting recognition evaluation,” *NIST 2004 Spring Rich Transcription Evaluation Workshop*, Montreal, Canada, 2004.

[4] Cohen, J., “The Gale Project: A description and an update,” *ASRU IEEE Workshop on Automatic Speech Recognition & Understanding*, 237-237, Dec. 2007.

[5] Paulsson, N., Choukri, K., Mostefa, D., Dipersio, D., Glenn, M. and Strassel, S., “A large Arabic broadcast news speech data collection,” *MEDAR 2nd International Conference on Arabic Language Resources and Tools*, Cairo, Egypt, 22-23 April, 2009.

[6] http://en.wikipedia.org/wiki/YouTube

[7] Barras, C., Geoffrois, E., Wu., Z and Liberman, M., “Transcriber: Development and use of a tool for assisting speech corpora production,” *Speech Comm.*, 33(1-2): 5-22, 2001.

[8] http://projects.ldc.upenn.edu/gale/Transcription/

[9] http://mandarin.about.com/od/chineseculture/a/dialects.htm

A2. M. Karnjanadecha and S. A. Zahorian, "**An Automatic Method for Determining Phonetic Boundaries for Continuous Speech Utterances in an Open Source Multi-Language Audio/Video Database**," *J. Acoust. Soc. America*, Vol 130.4.2, p2524, 2011, paper presented at the 162nd Meeting of the Acoustical Society of America, San Diego.

Nine hundred video clips (approximately 30 hours in each of English, Mandarin and Russian) have been collected from Internet sources such as youtube.com and rutube.ru. This multi-language audio/video database has been orthographically transcribed by human listeners with time markers at the sentence level. However, the aim is to provide this database to the public with high accuracy time markers at the phonetic level, which will greatly increase the value of the database. This paper describes an approach to achieving high accuracy automatic phonetic labeling based on a Hidden Markov Model speech recognizer. This automatic method was developed due to the great length of time and tediousness of performing this task using only human listeners. One major challenge for the automatic method was that the audio data consists of spontaneous speech with unconstrained topics and the speech was spoken under various acoustic conditions. The approach begins with a well-trained acoustic model for each language. The acoustic model is then adapted to each passage and finally the phonetic labeling of the passage is determined. Comparison of the automatically determined phone time markers with those obtained by human listeners, for a subset of the speech materials, shows the accuracy of the automatic method.

Words in abstract: 200

Technical area: Speech Processing and Communication Systems

(PACS) Subject classification numbers(s): 43.712Ar, 43.72Ne

No preference for lecture versus poster

Abstract of paper presented at the fall 2011 meeting of the Acoustical Society of America

A3 S. A. Zahorian and B. Wong, "**Spectral Amplitude Nonlinearities for Improved Noise Robustness of Spectral Features for use in Automatic Speech Recognition**," *J. Acoust. Soc. America*, Vol 130.4.2, p2524, 2011, paper presented at the 162nd Meeting of the Acoustical Society of America, San Diego.

Auditory models for outer periphery processing include a sigmoid shaped nonlinearity that is even more compressed than standard logarithmic scaling at very low and very high amplitudes. In some studies done at Carnegie Mellon University, it has been shown that this compressive nonlinearity is the most important aspect of the Seneff auditory model in terms of improving accuracy of automatic speech recognition in the presence of noise. However, in this previous work, the nonlinearity was trained for each frequency band of the Mel frequency cepstrum coefficients thus making it impractical to incorporate in automatic speech recognition systems. In the current study, a compressive nonlinearity is parametrically represented and constructed without training, to allow various degrees of steepness and "rounding" of corners for low and high amplitudes. Using this nonlinearity, experimental results for various noise conditions, and with mismatches in noise between training and test data, were obtained for phone recognition using the TIMIT and NTIMIT databases. The implications of the results are that a fixed compressive nonlinearity can be used to improve automatic speech recognition robustness with respect to mismatches between training and test data.

Words in abstract: 185

Technical area: Speech Processing and Communication Systems

(PACS) Subject classification numbers(s): 43.712Ar, 43.72Ne

No preference for lecture versus poster

Abstract of paper presented at the fall 2011 meeting of the Acoustical Society of America

A4. S. A. Zahorian , J. Wu, and M. Karnjanadecha, "**Non-Symmetric Time Resolution for Spectral Feature Trajectories**," *J. Acoust. Soc. America*, Vol 130.4.2, p2444, 2011, paper presented at the 162nd Meeting of the Acoustical Society of America, San Diego.

In a study presented at the fall 2010 meeting of the Acoustical Society of America (Zahorian et al., "Time/frequency resolution of acoustic features for automatic speech recognition"), we demonstrated that spectral/temporal evolution features which emphasize temporal aspects of acoustic features, with relatively low spectral resolution, are effective for phonetic recognition in continuous speech. These features are computed using Discrete Cosine Transform Coefficients (DCTCs) for spectral information from 8 ms frames and Discrete Cosine Series Coefficients (DCSCs) for their temporal evolution, over overlapping intervals (blocks) longer than 200 ms. These features are presented as an alternative to mel-frequency cepstral coefficients (MFCC), and their delta terms, for automatic speech recognition (ASR). In the present work, it is shown that these features are even more effective for ASR, using a non-symmetric time window which is tilted toward the beginning of each block when computing DCSCs. This non-symmetry can be implemented by combining two Gaussian windows with different standard deviations. This work also supports the hypothesis that the left context is somewhat more informative to phonetic identity than is the right context. Experimental results for automatic phone recognition are given for various conditions using the TIMIT and NTIMIT databases.

Words in abstract: 196

Technical area: Speech Processing and Communication Systems

(PACS) Subject classification numbers(s): 43.712Ar, 43.72Ne

No preference for lecture versus poster

Abstract of paper presented at the fall 2011 meeting of the Acoustical Society of America

APPENDIX B—DETAILED EXPLANATIONS OF RESEARCH ACTIVITIES

B.1 Detailed description of database

SUMMARY OF OPEN SOURCE MULTI-LANGUAGE AUDIO DATABASE

Stephen A. Zahorian and Montri Karnjanadecha
June 21, 2012

1. INTRODUCTION

To help support Spoken Language Processing Applications, a large database of web-hosted speech data was collected and annotated. More details can be found in [A-1]. Summarizing briefly, this data consists of approximately 30 hours of single speaker speech in each of three languages (English, Mandarin, and Russian). 300 recordings, averaging about 5 minutes in length, were collected for each language. This document summarizes some of the details not given in [A-1], such as directory structures and file types used for the database.

2. INITIAL FILE TYPES

1. Video—These files were downloaded from web sources, as YouTube , Youku, and RuTube, and immediately converted to high resolution MP4 format, and saved in this format.
2. Audio -- These were saved originally as 22.05k samples per second, stereo, 16 bits per sample, and saved as wav files (two's complement integers—no special coding). As described below, these files were also later converted to 16k mono wav files.
3. Transcriber files, orthographic transcriptions at the sentence or phrase level. All the words and noises in each sentence were transcribed by human listeners with the beginning and end of each sentence time marked. These files are saved as transcriber files (.trs), but are actually xml format.

3. FILES GENERATED AFTER PERFORMING FORCED ALIGNMENT

1) Waveform files of each passage (in 16k mono) with its corresponding time-marked phonetic-level, word-level and sentence-level transcriptions (HTK format). The 16k sample rate mono signals were created by averaging the L and R stereo channels and interpolating. Native English and Mandarin speakers looked at (with Cool Edit Pro software) and listened to left and right channels and to determine if there are some of these recordings, or sections of these recordings, where the simple averaging was not a good technique. It appears that for the vast majority of cases, the averaging was a good method, as both initial channels were the same or nearly the same. The Mandarin

listeners did note that invariably the R channel had a little more amplitude than the L channel. A spread sheet was created, summarizing possible issues with differences between L and R channels for each recording. Since the stereo versions have been saved, it would be possible to recreate mono versions in the future, if needed for any reason. In general the quality difference between 16k and 22k sampling rates is negligible, given the overall low quality and bandwidth of the original recordings.

2) Short waveform files (in 16k mono) segmented from each passages with their corresponding time-marked phonetic-level, word-level and sentence/phrase-level transcriptions (HTK format).

3) For each passage, 3 short segments were chosen for manual labeling. Thus, these 3 files have manually determined phonetic-level transcriptions (HTK format). Additionally, these three files have the automatic phone labels.

3. NOTES REGARDING LABELING/ TRANSCRIPTION FILES

3.1. Sentence level:

As mentioned above, sentence level transcriptions were obtained using human listeners and Transcriber software. These were originally transcribed at Binghamton University by students, with some corrections done by a second student for each passage, and checked again at AFRL. Another round of correcting/ editing occurred in the process of performing automatic forced alignment process. Issues such as correcting misspellings and typos, and creation of a custom word dictionary, were addressed, in order that the forced alignment could run without errors. During May and June of 2012, the forced alignment program was again run to correct any "glitches" in the trs files.

3.2. Word level:

Word level timing labels were created by the automatic forced alignment process (derived from the phone level transcription).

3.3. Phone level:

There are 2 sets of phone level labels: manual labels (obtained by manually labeling for a small portion of each video clip) and automatic labels (obtained by forced alignment software). Both types of labels are stored in different directories. Basically the manual labeling process starts the same way as for automatic labeling. Once we obtain automatic labels for each passage, we randomly selected 3 short segments for manual labeling. The automatic labels obtained were used as initial labeling. This greatly speeded up the phone boundary adjustment. An additional

step to complete the automatic labeling was to "merge" all phonetic level transcription files of short segment chunks to make one complete phone level transcription for each passage.

3.4. Directory structure:

For each language (English, Mandarin and Russian), 300 video clips were recorded. Each set of these clips are stored in a separate folder according to its language. Thus there are 3 main folders: \English for English clips, \Mandarin for Mandarin clips and \Russian for Russian clips. Each of these 3 main folders has a spread sheet, *.xls, which gives a brief summary of all the video clips for that language. There is also a second spread sheet with brief notes about the comparison of L and R channels for each recording.

Under \English folder, there are 300 subfolders (one for each clip) plus and additional subfolder called "\dict." The \dict folder contains 2 pronunciation dictionaries:

1. "cmudict07a_39.txt" is the standard CMU dictionary (CMU dict0.07a) which contains approximately 130k words.
2. "extra_wrd_dict_39.txt" is the additional dictionary which contains all extra words found in the database.

For each of the 300 "clip" folders, there are the following files and subfolders:

File or Subfolder	Description
C_name.mp4	Video clip, about 5 minutes long, in MPEG-4 format.
E_name_22k.wav	The audio file (22.05KHz sample rate, stereo) extracted from the video clip.
E_name.wav	The 16K mono version of the E_name_22k.wav. This file is used for forced alignment.
C_name.trs	Orthographic transcription of the audio data in Transcriber's XML file format.
\man_lab	Contains the 3 manual label files and 3 short waveform files (extracted from E_name.wav). Each wave files is approximately 5 seconds long.
\auto_lab	Contains all files generated by forced alignment.
\auto_lab\wav	Contains all short audio chunks segmented from the 16k version of the audio passage (E_name.wav). The segmentation boundaries are obtained from the sentence/phrases boundaries found in the transcription file. These short

	chunks were used, together with the word level label file in MLF format described in the HTK manual [A-2], in the actual automatic labeling process. Note that all intelligible speech and all noise segments extracted from the entire segment are included. The naming convention of the segmented files is described in the document "Phone Alignment Procedures."
\auto_lab\lab	Contains 6 automatically determined labels for the waveform E_name.wav. • E_name_phn.lab phone level label file (entire passage) • E_name_phn.mlf phone level label file (every short segment) • E_name_sen.lab sentence level label file (entire passage) • E_name_sen.mlf sentence level label file (every short segment) • E_name_wrd.lab word level label file (entire passage) • E_name_wrd.mlf word level label file (every short segment)

The following table shows the current status of the English database development.

Activity	Current status	
	150 Casual	150 Formal
Transcription file format verification	150	150
Misspelling words correction	150	150
Adding new words into dictionary	150	150
Manual labeling at the phonetic level	150	150
Automatic labeling at the phonetic and word levels	150	150

4. MANDARIN

The Mandarin database file formats, types of labeling, directory structure, etc., are essentially the same as for English, with these important differences:

1. The Transcriber manual transcriptions were created and saved using Mandarin character notation. These transcriptions were then automatically converted to PINYIN, and

subsequently manually corrected by human listeners. Thus the sentence level transcriptions are both in Mandarin character format and PINYIN (separate files). In PINYIN the tones are annotated as 0, 1, 2, 3, 4 for Neutral, High, Rising, Dipping, and Falling respectively. All automatic forced alignment was done at the phone level. The same general procedure for forced alignment was used as for English. Again, three short sentences from each passage were manually transcribed at the "syllable" level, as a ground truth for testing automatic methods. Note that all syllable level transcription files were saved as PINYIN only.

2. The dictionaries for Mandarin used in this work are different from those for English. Mandarin is a syllable-based language which has a limited number of syllables. The main dictionary for Mandarin was automatically generated to cover all possible syllables. 39 English phones were mapped to the pronunciation of each syllable. Since we had no Mandarin database to train the Mandarin speech recognizer, the TIMIT database was used. For this reason, the phonetic mapping from English to Mandarin was required. Similarly as for the English database, the secondary dictionary was also required to handle the out of vocabulary words. In many cases, English words were pronounced along with Mandarin speech. The pronunciations of these words are taken from the CMU dictionary that was used for English database. Both dictionaries use the same format as the CMU dictionary and have Pinyin notations.

4.1. Directory structure:

Under \Mandarin folder, there are again 300 subfolders (one for each clip) and a subfolder called "\dict." The \dict folder contains 2 pronunciation dictionaries:

1. "pinyin_dict_39.txt" is the English-to-Mandarin phone mapped pronunciation dictionary containing all possible Mandarin syllables (with tones).

2. "extra_pinyin_dict_39.txt" is another dictionary that contains extra words/syllables that are not found in the first dictionary. These words/syllables are mostly English words.

For each of the 300 folders, there are the following files and subfolders:

File or Subfolder	Description
C_name.mp4	Video clip, about 5 minutes long, in MPEG-4 format.
C_name_22k.wav	The audio file (22.05kHz sample rate, stereo) extracted from the video clip.

C_name.wav	The 16K mono version of the C_name_22k.wav. This file is used for forced alignment.
C_name_mandarin.trs	Orthographic transcription (using Mandarin characters) of the audio data in Transcriber's XML file format.
C_name.trs	Pinyin version of C_name_mandarin.trs. This file is used in further processing because the tools that we used do not support Mandarin characters.
\man_lab	Contains the 3 manual label files and 3 short waveform files (extracted from C_name_16.wav). Each wave file is approximately 5 seconds long.
\auto_lab	Contains all files generated by forced alignment.
\auto_lab\wav	Contains all short audio chunks segmented from the 16k version of the audio passage (C_name.wav). The segmentation boundaries are obtained from the sentence/phrases boundaries found in the transcription file. These short chunks were used, together with the word level label file in MLF format mentioned above, in the actual automatic labeling process. Note that all intelligible speech and all noise segments extracted from the entire segment are included. The naming convention of the segmented files is described in the document "Phone Alignment Procedures."
\auto_lab\lab	Contains 6 automatically determined labels for the waveform C_name.wav. • C_name_phn.lab phone level label file (entire passage) • C_name_phn.mlf phone level label file (every short segment) • C_name_sen.lab sentence level label file (entire passage) • C_name_sen.mlf sentence level label file (every short segment) • C_name_wrd.lab word level label file (entire passage) • C_name_wrd.mlf word level label file (every short segment)

The following table shows the current status of Mandarin database development.

Activity	Current status	
	108 Casual	192 Formal
Mandarin character to Pinyin conversion	108	192
Pinyin correction	108	192
Transcription file format verification	108	192
Adding foreign words into dictionary	108	192
Manual labeling at the syllable level	108	192
Automatic labeling at the phonetic and syllable levels	108	192

5. RUSSIAN

We have only the video files, the 22.05k stereo audio, and the "sentence level" transcriptions. These files are arranged in the same way as are the files for English and Mandarin, except only the files for the video, stereo audio, and manual transcriptions are included.

The following table shows the current status of the Russian database development.

Activity	Current status	
	91 Casual	209 Formal
Transcription file format verification	91	209
Misspelling words correction	0	0
Adding new words into dictionary	0	0
Manual labeling at the phonetic level	0	0
Automatic labeling at the phonetic and word levels	0	0

6. REFERENCES

[A-1] Zahorian, S. A. et al, "Open Source Multi-Language Audio Database for Spoken Language Processing Applications," Proc. INTERSPEECH-2011, pp.1493-1497, Florence, Italy, 2011.

[A-2] Young, S. J. et al, The HTK Book (for HTK Version 3.4), Cambridge University Engineering Department, 2006.

B.2 Phonetic Alignment Procedure

Montri Karnjanadecha and Stephen A. Zahorian

June 21, 2012

1. INTRODUCTION

This document gives a detailed explanation of automatic phonetic alignment method for the open source multi-language audio database (OSMLA). To make the OSMLA more useful, we have provided phonetic transcriptions for all speech files. Since manual labeling of speech data is time consuming and is not practical for this rather large database, an automatic method for automatic phonetic alignment has been developed and implemented. This document describes the method in detail.

2. TOOLS, RESOURCES, AND FILE FORMATS

2.1 Software tools and resources and typical uses

- Transcriber – Used to manually transcribe all audio passages at the sentence level
- Wavesurfer – Used to manually transcribe a small number of speech segments at the phonetic level (for English) and at the syllable level (for Mandarin).
- Online software used to convert transcription files in Mandarin characters to pinyin characters: http://www.mandarinbook.net/chinesetools/chinese_to_pinyin.php#output
- HTK tools – label file editing, dictionary management, HMM training and adaptation, HMM recognition, forced alignment
- Tfrontm – feature extraction
- Matlab – Matlab scripts were developed to automate all processes
- TIMIT – speech database used for building initial phone models
- Pronunciation dictionary (Lexicon) – CMU dictionary version 0.7a. This dictionary utilizes 39 English phones, and contains about 130k entries. It can be downloaded from: https://cmusphinx.svn.sourceforge.net/svnroot/cmusphinx/trunk/cmudict/

2.2 File formats

- Waveform – Windows file format (*.wav), PCM, 16 bits, uncompressed, stereo 22.05k stereo or 16k mono channel. (Note that 16k mono files were derived from 22.05k stereo by averaging stereo channels and interpolating.)

- Dictionary – Same standard as the CMU dictionary, no stress marks, allow entries with multiple pronunciations

- Transcription or label files – Initial transcription file that comes with each audio passage is in Transcriber's standard (XML format). During and after processing, all label files are in HTK format (plain label file or MLF (master label file) format).

3. TRANSCRIPTION VERIFICATION AND ADDING NEW WORDS TO DICTIONARY

As described in our Interspeech 2011 paper, [B-1], the last step before the automatic labeling step was to manually transcribe each audio passage at the sentence level with sentence boundary markers. The Transcriber software was used for this purpose. The transcription which contains all words was carefully checked and double-checked, but still there were typos and errors.

Each transcription file (in XML format) is analyzed to extract noise events, words, special symbols, speaker ID, time markers and other useful information. At this stage, a list of all unique words is of main concern. From the word list, we can check if each word has its pronunciation in the dictionary. There are 2 reasons which cause the pronunciation to be missing. The first reason is the word is misspelled and the second reason is that the word is out of vocabulary. For the out of vocabulary case, the word could be a foreign word, acronym, abbreviation, proper noun, jargon, etc. We added a new entry to the dictionary so that this particular word can be recognized. Table B-1 shows examples of words that are not in the CMU dictionary.

Table B-1. Examples of words not in the CMU dictionary, but appearing in OSMLA

Word	Pronunciation	Description
ACRO-	AH K R AO	Unfinished or disfluency
IPHONE'S	AY F OW N Z	proper noun/ jargon?
MATLAB	M AE T L AE B	proper noun
[__ALBRIGHT]	AO L B R AY T	unclear word
[~HTML]	EY CH T IY EH M EH L	Acronym
LANG_ES_ESPANA	EH S P AH N Y AH	Foreign word

In practice, we did not augment the CMU dictionary with new entries, but we created a new dictionary for these out-of-vocabulary (OOV) words. The tools that we used can logically combine these 2 dictionaries at run time.

To extract OOV words from each transcription, we analyze the transcription file using a Matlab script ("split_wav_complete.m"). The outputs of this step are:

1) sequence of sentences/phrases with time markers

2) sequence of words, short pauses, and noise events for each sentence/phrase

3) number of speakers

4) name of each speaker

From each sequence of words we determined if each word in the sequence was in the dictionary. If an OOV word was found, we inspected the transcription file to see if the word was misspelled or not. All misspelled words were corrected. The remaining OOV words were added to our dictionary with their corresponding phonetic realization. To speed up this step and to make sure that a phonetic realization of each OOV word was correct, we reused the pronunciation from the CMU dictionary as much as we could. For example the word "iPhone" is pronounced as the word "PHONE" preceded by the "AY" sound. If part of any word had multiple pronunciations in the CMU dictionary, we also put all possible alternative pronunciations in the dictionary.

For the case of a foreign word, we have to listen to the word from the clip and try as best as we can to give the phonetic transcription of the word using the "standard" 39 English phones.

Acronyms, such as USB, DVD, CD, and CEO, occur very often in our recorded video clips. Such words were transcribed with the "~" prefix which makes them appear as: ~USB, ~DVD, ~CD, and ~CEO. Please see [B-1] for details. For example, we can differentiate that the word CAD has to be pronounced as K AE D and the word ~CAD as S IY EY D IY.

Unfortunately, HTK tools cannot work correctly with a dictionary entry having the "~" as the first character--thus we put a square bracket around every acronym to avoid this problem. For example, the word ~USB found in the transcription file is referred to as [~USB] in the dictionary.

A foreign word such as ESPANA in Spanish is transcribed in the transcription file with a "language" tag. Such a tag is identified during the transcription file analysis. The "language" tag has a language identifier, which could be "FR" for France, "IT" for Italian, "ES" for Spanish, etc. Dictionary entries for each foreign word is preceded by "LANG_XX_," where XX is the language identifier. For example the word ESPANA in Spanish will be found in the dictionary as "LANG_ES_ESPANA".

Unclear words were transcribed inside double parentheses. For example, if we found ((say)) in the transcription, it means that the human listener was not sure about the word he/she heard. All unsure words were preceded by "__" and were put inside a square bracket. For example the word ((Say)) will look like "[__SAY]" in the dictionary, with its pronunciation (phonetic realization) exactly the same as for the word "SAY."

4. AUTOMATIC LABELING USING FORCED ALIGNMENT

4.1 Preliminary experiments

Several experiments were conducted to find the best technique for our task, which is aligning a database consisting of real-world speech. This speech is understandable by human listeners, but quite low quality, due to both noise and distortion; it also un-rehearsed, and spontaneous, for the most part. Most reported work for forced alignment is based on experiments with clean, high quality, read speech. Most of the available techniques cannot be directly used effectively with our database. As documented in [B-2], the best method for automatic phonetic alignment of our database is to use a HMM-based speech recognizer running in the forced-alignment mode. The best way (of several methods tested) to train the HMM models is to start with TIMIT models, and adapt to each passage. Each time the model is adapted, the forced alignment is performed. Thus this is a passage-by-passage forced alignment.

4.2 Steps

i) Train HMM models using TIMIT corpus

ii) Preprocess the waveform and transcription files

This step has 3 sub-steps:

a) Parse the transcription

b) Divide the waveform into short segment according to sentence/phrase markers

c) Create word level transcription for each short segment in HTK's MLF format

iii) Generate phone level transcription from word level transcription using the HLed tool and the pronunciation dictionaries.

iv) Adapt TIMIT models to each passage using the transcription in 4.2.3

v) Force align the phone level transcription using the adapted models.

vi) Use the forced aligned transcription to adapt the HMM models

vii) Repeat steps 4.2.5 and 4.2.6 for several times

viii) Connect the phone level transcriptions of all short segments

4.3 Training HMM models using TIMIT corpus

The original TIMIT phone set consists of 61 phones. Since the CMU pronouncing dictionary that we use employs 39 phones, we collapse the TIMIT phone set to 39 phones + silence. The TIMIT HMM models were created and trained using the HTK tools. We used a Matlab script called HTKtool.m to automate all training processes. This script was initially developed by Hongbing Hu and was extensively used for our phonetic recognition experiments. Table B-2 summarizes some important parameter settings.

Table B-2. Some important parameter settings

Parameter	**Typical Values**	**Remark**
# of states per HMM	4	
# of Gaussian mixtures per state	32	
# of HMM models	40	39 phones + silence
# of iteration to run HInit	20	
# of iteration to run HRest	20	
# of iteration to run HERest	8	
Training data	All TIMIT training set	
Feature set	78 DCTCs/DCSCs	"Best" setup
Tfrontm version	76.4	

Table B-3 shows the list of 39 phones + silence.

AA	AE	AH	AO	AW	AY	B	CH
D	DH	EH	ER	EY	F	G	HH
IH	IY	JH	K	L	M	N	NG
OW	OY	P	R	S	SH	T	TH
UH	UW	V	W	Y	Z	ZH	SIL

4.4 Preprocess the waveform and transcription files

The Matlab script called "split_wav_complete.m" was written and used to automate this preprocessing step. It was designed to work correctly either with English or Mandarin (Pinyin) transcription. The script has the following interface:

[error, num_speakers, num_foreign_words] =

split_wav_complete(in_trs_file, wav_in_dir, wav_out_dir, lab_out_dir, new_fs, save_org_wav)

Table B-4 summarizes the meaning and function of each input/output variable.

Table B-4. Meaning and function of each variable

Variable name	**Type**	**Description**
in_trs_file	input	Name of the transcription file (a file with .trs extension)
wav_in_dir	input	Directory name that stores the corresponding waveform file. Once the program has opened the transcription file, it will search for the corresponding waveform file in this folder. Note that the transcription file and its corresponding waveform file must have the same name (but different extension).
wav_out_dir	input	Name of directory to store the resulting waveform files after segmentation.
lab_out_dir	input	Name of directory to store the resulting label files after analyzing the transcription file.

new_fs	input	Desired sampling frequency. Originally all waveform files are in 2-channel 22.05kHz sample rate. The waveform file under processing will be resampled to the new_fs and also converted to mono channel.
save_org_wav	input	Set to 1 if we want to store the resampled (and mono) version of the waveform
error	output	This return value is non-zero if any error occurred
num_speakers	output	Number of speakers presented in the passage. Some passages were spoken by multiple speakers. Currently, this return value is always ignored.
Num_foreign_words	output	Number of foreign words presented in the passage. Currently not used.

This program reads the transcription file and parses the text in the file. As discussed in a previous section, some results from parsing the transcription file are the sequences of words of each sentence/phrase and each sentence/phrase time markers. The sentence/phrases time markers (in milliseconds) are used to segment the waveform file into several short chunks. It is advantageous to segment the long waveform (~5 minutes) into short segments because 1) we can ignore any unwanted audio segment and 2) this would be more accurate to automatically align a short segment than a very long one.

The following is a snapshot of the content of a transcription file. This example illustrates how we parse the transcription and make use of the obtained information.

```
<Sync time="30.112"/>
Umm
<Event desc="p" type="noise" extent="instantaneous"/>
 but I'm going to start off with my favorite. So.
<Sync time="33.229"/>
```

The “Sync time” tags indicate start and end times of this audio segment. The information obtained after parsing this snapshot are:

ix) start time = 30.112 seconds

x) end time = 33.229 seconds

xi) Text: “Umm [P] but I’m going to start off with my favorite. So.”

Note that the symbol "[P]" in the word sequence denotes the "short pause" event.

The start time and end time are used to segment the original audio passage (into a short chunk) at the right location.

In summary, the program parses the transcription file, loads the corresponding waveform file, down-samples the waveform file, segments the waveform file into several short chunks, and generates the word level transcription file for each short chunk.

Approximately 100 short waveform files are segmented from each audio passage. Each of the waveform files has its corresponding word level transcription, as obtained from manual transcriptions. To reduce the number of files, we store all word level transcriptions in an MLF file. The MLF file, used by HTK, is a formatted text file which can hold multiple label files. Please see the HTK manual [B-3] for more details.

In the beginning every word in the word level transcription has the begin time and end time equal to 0. After automatic alignment, each word will get its "predicted" begin and end times. However the begin time of the first word will always starts from 0, which means the time indices are relative to its short audio segment. To obtain the "right" time labeling for the entire audio passage, we have to connect or stitch all the transcriptions back together in the right order. Thus when we segment the audio passage, we keep track of the order of each audio segment by augmenting a 3-digit running number into its file name. For example for the passage "E_CC_EFAD01_01" has its transcription file "E_CC_EFAD01_01.trs" and its waveform files "E_CC_EFAD01_01.wav." After this preprocessing step the resulting waveform files are:

```
E_CC_EFAD01_01_001_000000_000379_SIL.wav
E_CC_EFAD01_01_002_p_000379_013151_CLEAN1.wav
E_CC_EFAD01_01_003_013151_013534_SIL.wav
E_CC_EFAD01_01_004_p_p_ot_013534_023296_DIRTY.wav
E_CC_EFAD01_01_005_023296_023470_SIL.wav
E_CC_EFAD01_01_006_023470_029669_CLEAN0.wav
E_CC_EFAD01_01_007_029669_030112_SIL.wav
E_CC_EFAD01_01_008_p_030112_033229_CLEAN1.wav
E_CC_EFAD01_01_009_033229_033849_SIL.wav
.
.
.
```

The first 14 characters representing the source file name are the same for all files. Detailed explanations of the file name convention for source files can be found in [1]. The remaining characters in the file name also convey information. This is explained in the following example.

e.g.

E_CC_EFAD01_01_004_p_p_ot_013534_023296_DIRTY.wav

E_CC_EFAD01_01 -- Passage name

_004 -- a running number starting from 001 to an ending number

_p_p_ot -- list of noise events found in this short segment, vary from segment to segment, can be empty

_013534_023296 -- start time and stop time in ms. It indicates where in the waveform file that this segment is located.

_DIRTY -- category of segment. This could be “SIL” , “NOISE”, “DIRTY”, “CLEAN0”, and “CLEAN1”.

Table B-5. Detail description of each segment category

Category	**Description**
SIL	Segment contains only silence (short pause)
NOISE	Segment contains only a noise or a sequence of noises (no speech)
CLEAN0	Segment contains only clean speech, no short pause, no instantaneous noise events or background noise.
CLEAN1	Segment contains only clean speech and short pauses
DIRTY	Segment contains speech and noises (noisy speech).

Embedding some information into a file name is useful and convenient for some steps. For example at some stage of our experiments, only clean speech was used for HMM training/adaptation. Thus we only selected files whose names ended with “CLEAN?” for processing. However it is not very useful in the end and it makes this looks too complex, potentially even confusing. Extra information should be removed before distributing through the LDC.

The following is the first few lines of the MLF file generated in this step.

```
#!MLF!#
```

```
"*/E_CC_EFAD01_01_001_000000_000379_SIL.lab"
0 0 [P]
.
"*/E_CC_EFAD01_01_002_p_000379_013151_CLEAN1.lab"
0 0 HELLO
0 0 LOVELIES
0 0 [SP.]
0 0 WELCOME
0 0 BACK
0 0 SO
0 0 I'M
0 0 GOING
0 0 TO
0 0 BE
```

Note again that, the time indices of every word in the transcription are initially zeros. These indices will get their values after the automatic alignment process has been completed.

4.5 Generate phone level transcription from word level transcription

Since the HMM models that are used for forced alignment of the audio data are phone models, the word level transcriptions are not directly usable. A pronunciation dictionary can be used to convert a word sequence into a phone sequence. The HTK's HLed tool is used to convert a word level label file into a phone level label file. Note that some words may have multiple pronunciations, but we ignored alternate pronunciations in work to date (June , 2012). At this stage we just used the first pronunciation found in the dictionary.

4.6 Adapt TIMIT models to each passage using the transcription

All 40 TIMIT phone models are adapted to each passage using the HTK's HERest tool. Inputs needed for this step are:

1) Feature files extracted (the same way as did for TIMIT data when training the TIMIT model) from each waveform file.

2) TIMIT phone models

3) Phone level transcription for each feature file.

4) List of all names of phone models (referred to as "hmmlist" in HTK manual)

Note that any waveform of shorter than 0.5 seconds is not included in the adaptation data because it has too few frames of data and usually causes a problem with HMM adaptation/evaluation.

4.7 Forced alignment

In this step, the adapted models are used to align the waveform with phonetic labels and time markers. The HTK's HVite tool is used for this purpose. The HVite tool is configured to run in the forced alignment mode, which requires that the transcription must be provided to the tool. Word level transcriptions and a pronouncing dictionary are provided to the HVite tool at this step. The HVite tool can choose the most likely pronunciation for words with multiple pronunciations. The results from this step are the phone level transcriptions with time markers for every short audio segment.

The same set of the adaptation data is used for forced alignment.

4.8 Improving alignment accuracy using an iterative process

The phone level transcriptions obtained with forced alignment are used to adapt the TIMIT model again to improve the alignment accuracy. This process can be repeated several times.

4.9 Connect the phone level transcriptions of all short segments

This step tries connects each phone level transcription for short audio segments together to make a single completed transcription for the entire audio passage. Some short segments such as those with background noise only are not used for HMM adaptation and have no automatic alignment transcription. However the time labeling information of such segments can be obtained easily from the time indices encoded in its file name.

As noted in section 4.6 and 4.7, very short waveforms are not used in adaptation and forced alignment. Thus there is no time information for such waveforms. To complete the label connection, we need to manually align these missing files. Although there are quite a large number of incomplete files, most of them can be fixed automatically. For example a file with single label (such as a pause, a noise event) can be automatically labeled by just using its start time and stop time available from its file name.

5 Garbage Model

Noises such as human noise, laugh, music, animal noise, nature noise, vehicle noise, machine noise, office noise, and other noise are modeled as "garbage." The garbage model is initially cloned from TIMIT's silence model. It is later adapted to noises. All sorts of noised are collapsed to a single garbage model. The pronunciation dictionary is modified to included pronunciations of all noises with a map to the garbage model.

References

[B-1] Zahorian, S. A. et al, "Open Source Multi-Language Audio Database for Spoken Language Processing Applications," Proc. INTERSPEECH-2011, pp.1493-1497, Florence, Italy, 2011.

[B-2] Karnjanadecha, M., and Zahorian S. A., "Toward an Optimum Feature Set and HMM Model Parameters for Automatic Phonetic Alignment of Spontaneous Speech, " To appear in Proc. Interspeech 2012.

[B-3] Young, S. J. et al, The HTK Book (for HTK Version 3.4), Cambridge University Engineering Department, 2006.

B.3 Forced Alignment Software User's Guide

Montri Karnjanadecha and Stephen A. Zahorian

June 21, 2012

1. INTRODUCTION

An HMM-based forced alignment system was used to automatically determine the boundaries of speech signals at the phonetic level for English speech and at the syllable level for Mandarin speech. The HMM was implemented with the HTK tools version 3.1. Several Matlab scripts were written to automate all steps. All scripts were developed on Matlab R2010a and used on Matlab R2010a and Matlab R2011a.

During the process of building acoustic models for forced alignment (which is the same as for speech recognition), several HTK tools were executed. For example the HLed tool was used for editing label files and the HERest tool was used for HMM training and adaptation. The basic approach to this is to invoke HTK tools, which come in the form of *.exe files, under the dos-like "command prompt." Note that in most cases, running a HTK tool requires several command line arguments. This is very inconvenient, especially when each tool must be run with different options.

Matlab has a very useful function called "system." This function allows us to call any executable file or any shell command from inside Matlab's environment. We can write a Matlab script to serve as a wrapper function which calls one or more HTK tools. This method has greatly improved the flexibility and simplicity for calling HTK tools. As a result, many Matlab functions were developed to support the forced alignment process. Forced alignment can be performed in a batch mode with various options without having to type in a command line.

Although use of Matlab scripts allow us to perform forced alignment by executing only one Matlab script, one of the major problems is that any user who is not familiar with the HTK tool and the forced alignment process will still have difficulties when using these Matlab scripts, in isolation. For example, to compute speech features, we must have 1) list of waveform files, 2) setup file for TFRONT.m (our in-house feature computation program) or for Hcopy.exe (HTK's feature computation tool), and 3) list of resulting feature files with a target directory. Preparing all this data can be troublesome. Another very difficult example would be to perform HMM model adaptation. To adapt HMM models we need source models, several setup files for HERest.exe tool, input feature files and other supporting files. It would not be possible for most general users to directly use the Matlab scripts for this rather complex task.

The software described in this document combines these scripts with a user-friendly GUI, making the overall process much easier to perform. Note that we hired undergraduate student

assistants who had limited knowledge on speech processing and using HTK tools. Providing them with an easy-to-use program has greatly speeded up the detailed labeling process.

FA_7.m is a Matlab program that integrates all Matlab scripts required to complete the forced alignment task. Note that FA stands for Forced Alignment. It utilizes a graphic user interface (GUI) which is very easy to use. Anyone can use the program without having to understand the details of its internal processes. Figure C-1 shows a screen shot of this program. The buttons on the left half of the figure are used to execute functions. The edit boxes on the right half of the figures show some control variables whose values can be changed by the user at any time; these values can also be changed by modifying a setup file.

This document explains how to use the FA_7.m program. It also briefly describes the program's internal functions which could be useful for anyone who wants to modify the program.

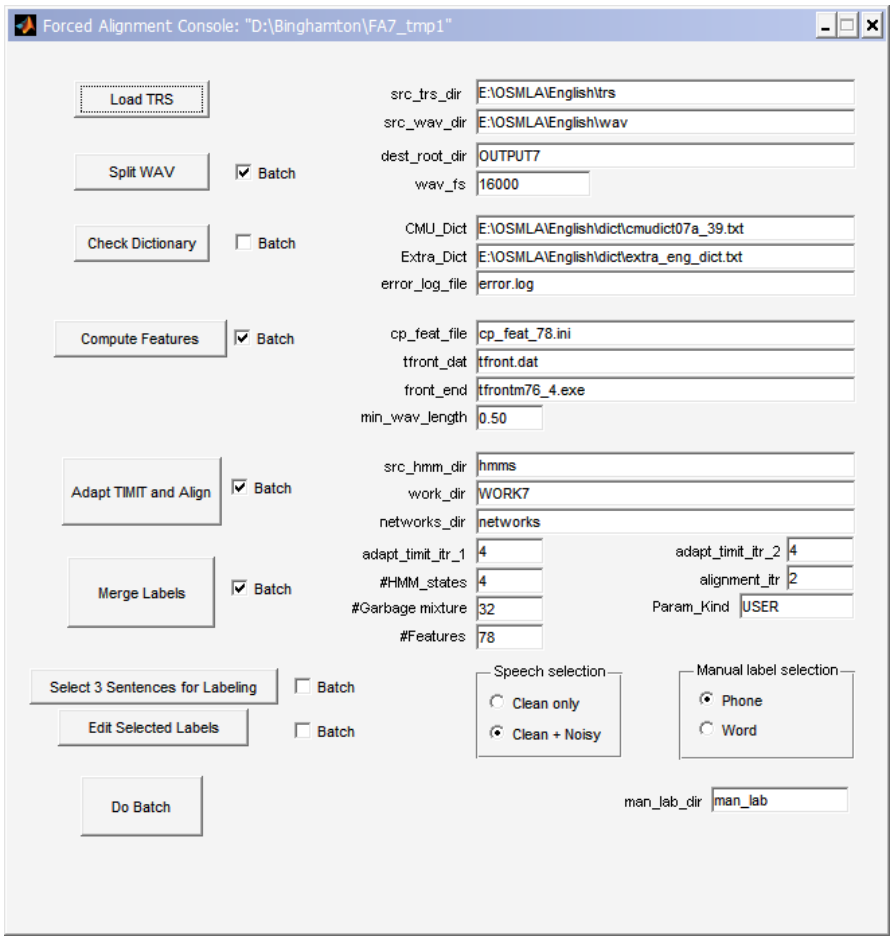

Figure C-1. A screen shot of the forced alignment program

2. FUNCTIONALITY

The main function of this program is to automatically align speech signals at the phonetic, word or syllable level. The program is designed to work on either English or Mandarin speech. Note that the orthographic transcription for the Mandarin speech must be in Pinyin notation. HMM models of 39 English phones were trained using TIMIT data. They later were adapted to each passage. The alignment is achieved by running the HMM recognizer in the forced alignment mode. In addition this program can be used to speedup the manual labeling process by providing initial phonetic boundaries.

3. SETTING UP

This program calls several HTK tools, each of which is an executable file. By default the HTK tools are stored under htk\bin.win32\ folder. The system path must include this folder so that the HTK tools can be invoked from any location.

Many short Matlab scripts were developed and used to run this forced alignment program. They are stored in the folder called Matlab_common\. It is suggested to include this folder in the Matlab path so that we can have only one copy of these codes and any script under this folder can be executed in Matlab without having to change the current directory to match each folder.

If Tfrontm is used for feature computation, either Tfrontm.m or TFrontm.exe (an execution version of TFrontm.m) will be required. If Tfrontm.m is used, Matlab's path must point to all associated *.m files.

4. USING THE PROGRAM

After all setup variables have been set, using this program consists of just pushing each button in the correct order. This section explains functionality of each button from top to bottom. The explanation is organized in Table C-1.

Table C-1: Function of each button

Button	Function
Load TRS	This will open a file dialog. The user has to select 1 or more *.trs files. *.trs is a transcription file obtained with the Transcriber program. Each audio passage (in waveform file *.wav) has a corresponding transcription file. The transcription file was created manually. It contains the orthographic transcription of the speech in the audio passage, with sentence boundaries. The code was written such that multiple TRS files can be selected at once. This allows us to perform forced alignment on multiple files with a single run.
Split WAV	An audio passage can be 3-6 minutes long. Thus it would be better to segment each audio passage into shorter segments. Using sentence boundary information from the transcription file, we can split the original waveform file into several short waveform files. Each short file has its corresponding transcription file. When this function is executed, the loaded TRS file is analyzed and the waveform is split. If there is any error in the TRS file the program will terminate. The Matlab script "split_wav_complete.m" is used to perform this function.
Check Dictionary	This function is used to check if every word in the passage is in the pronunciation dictionary. If the pronunciation of a word is missing from the dictionary, the program will stop and the user must add the pronunciation of the word into the dictionary.
Compute Features	This function takes short waveform files from the previous step as input and computes features. It supports TFRONTM feature calculations and HTK's HCopy tools.
Adapt TIMIT and Align	This is the most complex operation. It reads TIMIT models (HMM's that have been trained with TIMIT data), then performs model adaptation and forced alignment. The adaptation and forced alignment is repeated several times to improve alignment accuracy. This step employs several HTK tools and there are several Matlab scripts that support this function.
Merge Labels	This step merges all transcription file obtained from the forced alignment step to create a single transcription file. The output transcription file is stored in HTK format. This is the final step of forced alignment.

Select 3 Sentences for Labeling	For each passage, we manually labeled 3 sentences. These manually aligned labels can be used as "ground truth" information for algorithm development. When this button is pressed, the program will systematically select 3 sentences.
Edit Selected Labels	This button is used to invoke WaveSurfer program to manually label the 3 selected sentences.
Do Batch	If this button is pressed, any above button that has the Batch box checked will be executed in order from top to bottom. This allows us to run every step with a single click.

5. CONFIGURATION VARIABLES

There are a number of configuration variables that control the behavior of the forced alignment program. Table C-2 summarizes the meaning of each variable. The value of each variable can be changed by modifying the FA_7_setup.m script. Note that these variables can be changed during runtime so that the change can take effect immediately.

Table C-2. Meaning of each configuration variables

Variable	**Meaning/Function**
src_trs_dir	Directory where the transcription files (*.trs) are stored.
src_wav_dir	Directory where the waveform files (*.wav) are stored.
dest_root_dir	Root directory to store all output files. Each subdirectory under this directory has the same name as the speaker's name.
wav_fs	Sampling frequency of output waveform. The original waveform is in 22kHz sampling rate. Usually this variable is set to 16000.
CMU_dict	Main dictionary file. For English, this should be CMUdict 0.07a without stress marks. For Mandarin, this should be a word (syllable) dictionary which is automatically generated by implementing Mandarin pronunciation with the English 39

	phone set.
Extra_dict	Additional dictionary file. Same format as CMU_dict. User is supposed to add new words to this dictionary instead of to the main dictionary. This dictionary will get sorted every time when the automatic alignment is executed.
error_log_file	Log file to store error messages
cp_feat_file	Configuration file used for feature extraction, usually set to "cp_feat78.ini" for TFRONTM or "hcopy.conf" for HTK's HCopy
tfront_dat	Tfront.dat file for TFRONTM. Not used for HCopy.
front_end	TFRONTM.EXE, TFRONTM.M or HCOPY
min_wav_length	Minimum length of waveform in seconds which the program considers. Waveform files of shorter than this length will be ignored.
src_hmm_dir	Source directory that stores all TIMIT HMM models.
work_dir	Working directory, to store all temporary files; can be removed after finished.
networks_dir	Directory to store phone network or syllable network. This is needed for HTK's HVite tool. For the current version of the program, this directory is used to store some temporary files.
adapt_TIMIT_itr_1	Number of iterations to adapt the TIMIT model (first pass). This is the number of iterations to run embedded re-estimation (HERest). Should be set to 4 at least.
adapt_TIMIT_itr_2	Number of iterations to adapt the TIMIT model (second pass). This is the number of iterations to run embedded reestimation (HERest). Should be set to 4 at least.
#HMM_states	Number of states in each HMM model. This number is the number of states in TIMIT models and adapted models. It works best with 4 states per HMM.
alignment_itr	Number of iteration to repeat the entire forced alignment process. More iterations yield higher accuracy but too many iterations are not very helpful. 2 iterations will be ok, 4 will be

	better.
#Garbage_mixture	Number of mixtures used in building a garbage model. All noise events are pooled together and modeled as garbage.
Param_Kind	Type of features. When HTK reads a feature file, it must know the feature type so that is can handle the contents properly. If TfrontM is used to compute feature, this variable must be set to USER. If we used HTK features such as MFCC, LPCC, etc, this variable should be set accordingly.
#Features	Number of features used for building TIMIT models. This same number has to match with cp_feat_file (cp_feat_78.ini) or HCopy.conf. The same number and the same feature calculation method must be used to compute feature for TIMIT models training, and HMM adaptation for forced alignment.
Speech Selection	Use only clean speech or clean+noisy speech for model adaptation and alignment. The former setting can be used if we want to select only clean speech for adaptation and alignment. This may fail on some noisy passages. The later setting will always work. It take all available data (except for too short segment) to perform model adaptation and forced alignment.
Manual Label Selection	To perform forced alignment at phonetic level for English, choose "Phone." Select "Word" to align at the word or syllable level (for Mandarin). Internally, this program aligns at the phonetic level. However, the user can specify the final output.
man_lab_dir	Directory to store manual labels. If we click the "Select 3 Sentences for Labeling" the selected label files and their corresponding waveform files will be copied from the "work_dir" to this directory. User can easily access to this folder. When the "Edit Selected Labels" button is clicked, the WaveSurfer program will be invoked and all files in this folder will be open.

6. CODE LOCATION

All necessary codes for this software can be found from ZahorianResearch\Montri\Code. The codes are stored in 3 folders: Forced_Align_GUI_7\, Matlab_common\, and TFRONTM\.

The main program, "FA_7.m," and other supporting files are stored under Forced_Align_GUI_7\ folder. FA_7.fig is the GUI portion of the main program. This file can be opened and edited using the Matlab's GUIDE tool.

Other Matlab scripts are stored in Matlab_common\ folder. These scripts are called by the main program. Matlab path must also point to this folder.

List of symbols, abbreviations and acronyms

AFRL	Air Force Research Laboratory
ASR	Automatic Speech Recognition
DCT	Discrete Cosine Transform
DCTC	Discrete Cosine Transform Coefficient
DCS	Discrete Cosine Series
DCSC	Discrete Cosine Series Coefficient
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
LDC	Linguistic Data Consortium
MFCC	Mel-Frequency Cepstral Coefficient
NN	Neural Network
OGI	Oregon Graduate Institute