Robust Long Speech-Text Alignment

A Project Report

submitted by

Eddula Dileep Kumar Reddy
(EE13B024)

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2017

THESIS CERTIFICATE

This is to certify that the thesis titled **Robust long speech-text alignment**, submitted by **Eddula Dileep Kumar Reddy**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. S. Umesh
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600036

ACKNOWLEDGEMENTS

This work would not have been possible without the guidance and the help of several people. I take this opportunity to extend my sincere gratitude to all those who made this thesis possible.

First, I would like to thank all my teachers who bestowed me with good academic knowledge. I am indebted to my advisor **Prof. S. Umesh** whose expertise, generous guidance and support made it possible for me to work on a topic that was of great interest to me. I would also like to thank my lab mate and dear friend **Srinivas** for sharing his valuable ideas and helping me whenever I am stuck with some problem. A special thanks to **Pothumahanti Srikar**(my project partner) and **Sandeep Kothinti** who had helped in understanding a completely new field to me.

ABSTRACT

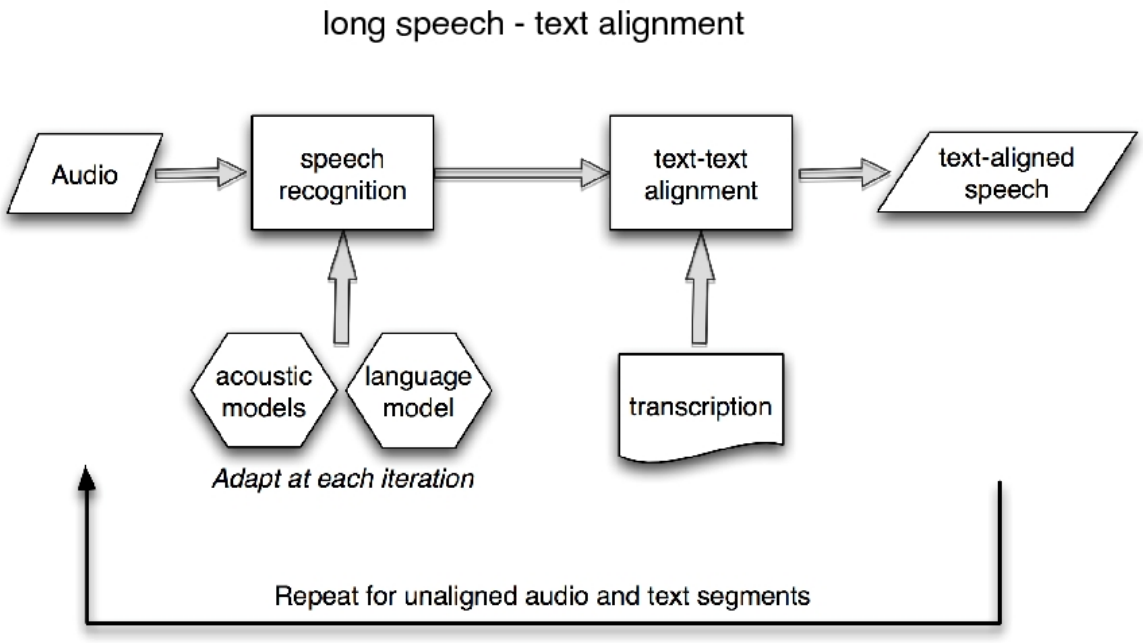
Long speech-text alignment can facilitate large-scale study of rich spoken language resources that have recently become widely accessible, e.g., collections of audio books, or multimedia documents. For such resources, the conventional Viterbi-based forced alignment may often be proven inadequate mainly due to mismatched audio and text and/or noisy audio. In this paper, we present a kaldi based project which is a software toolkit for robust long speech-text alignment that circumvents these restrictions. It implements an adaptive, iterative speech recognition and text alignment scheme that allows for the processing of very long (and possibly noisy) audio and is robust to transcription errors.

Contents

Chapter 1

Introduction

Speech-text alignment commonly finds applications in fields such as multimedia indexing and training of large vocabulary speech recognition and synthesis systems. Recently, it has also been shown to be useful in the context of phonetics research for the exploitation of rich spoken language resources such as audio books. Overall, it may be viewed as a mechanism that simultaneously enriches spoken language transcriptions with temporal information and identifies audio segments with their corresponding spoken content. Conventionally,speech-text alignment is performed by application of the standard Viterbi-based forced alignment. However, this process may be proven inadequate in cases when the audio is contaminated with noise or when the transcription is not sufficiently accurate. In this paper, we present our Kadli based software, that circumvents these restrictions to a significant degree through the implementation of an adaptive, iterative speech recognition - text alignment scheme as shown in below figure.



The robust long speech-text alignment approach implemented builds upon the iterative segmental application of a large vocabulary continuous speech recognition system, as introduced by Morenoand his colleagues. The main idea is based on the assumption that with a sufficiently good speech recognition engine, it is possible to pose the speech-text alignment problem as a text-text alignment one.Solution of the latter is normally much less computationally demanding. After selecting regions where alignment is reliable, based on prescribed criteria, the process is iterated on the remaining unaligned regions. Language modeling becomes region specific, resulting in improved recognition and consequently improved alignment.Finite state automata have been used to account for insertions, deletions,and/or substitutions of the transcribed words. We also implemented finite state grammar approach at the final stage to appropriately constrain the alignment search space if necessary.We improved robustness by label boosting i.e., adaptation of the acoustic models at every iteration to account for mismatched acoustic conditions.

1.1 Dataset

Sail Align is open-software which does the same task which we are doing but it only implements in English and some other international languages, the main task of this project is to implement same in Hindi so we decided to take some corpus of Hindi audio and text files

The data set we used for this project is the modi's mann ki baat speech of every month (http://www.narendramodi.in/mann-ki-baat). Initially we had a data set of thirty minutes each audio and corresponding text file but to build a generic acoustic model we had to break these audio files into small chunks of around 1 minute each and also there corresponding text files. We used audicity software for this task and got a 3 hours of data set. The data set we worked is in Hindi but some of the words were in English. We had to translate them into Hindi for which we used a website - (http://www.quillpad.in/editor.html).We also had to remove all the punctuation marks in the text data.Here for both training and test data the speaker is unanimous and there is no much of noise robustness to get a good working model we have to have a good data with different speakers and to improve noise robustness we have implemented Maximum Likelihood Linear Regression (MLLR).

Training data - The 3 hours of data set which we have chucked into small segments

Test data - A new modi speech file about 30 minutes and text file in which all the punctuations are removed and English are translated

1.2 Building Lexicon

For building Lexicon we used **Unified** software developed by IITM (you can find it in donlab iitm)

Files we build using unified

- **lexicon.txt**
- **nonsilence phones.txt**
- **optional silence.txt**
- **phones.txt**
- **silence phones.txt**
- **text.txt**

Chapter 2

Kaldi and Generic Acoustic Model

2.1 Kaldi

Kaldi, a free, open-source toolkit for speech recognition research. Kaldi provides a speech recognition system based on finite-state transducers (using the freely available Openfst), together with detailed documentation and scripts for building 16 complete recognition systems. Kaldi is written is C++, and the core library supports modelling of arbitrary phonetic-context sizes, acoustic modeling with subspace Gaussian mixture models (SGMM)as well as standard Gaussian mixture models, together with all commonly used linear and affine transforms. Kaldi is released under the Apache License v2.0, which is highly non-restrictive, making it suitable for a wide community of users. The goal of Kaldi is to have modern and flexible code that is easy to understand, modify and extend. Kaldi is available on SourceForge. The tools compile on the commonly used Unix-like systems and on Microsoft Windows.

2.1.1 Features Extraction

Automatic speech recognition (ASR) has made great strides with the development of digital signal processing hardware and software. But despite of all these advances, machines can not match the performance of their human counterparts in terms of accuracy and speed, specially in case of speaker independent speech recognition. So today significant portion of speech recognition research is focussed on speaker independent speech recognition problem. The reasons are its wide range of applications, and limitations of available techniques of speech recognition.

Mel Frequency Cepstrum Coefficients (MFCC)

Mel frequency Cepstral coefficients algorithm is a technique which takes voice sample as inputs. After processing, it calculates coefficients unique to a particular sample.MFCC takes human perception sensitivity with respect to frequencies into consideration, and therefore are best for speech/speaker recognition.MFCCs being considered as frequency domain features are much more accurate than time domain features. MFCC is an audio feature extraction technique which extracts parameters from the speech similar to ones that are used by humans for hearing speech, while at the same time, deemphasizes all other information. The speech signal is first divided into time frames consisting of an arbitrary number of samples. In most systems overlapping of the frames is used to smooth transition from frame to frame. Each time frame is then windowed with Hamming window to eliminate discontinuities at the edges. In this project we have used this time frame as 10 msec. Kaldi feature extraction and waveform-reading code aims to create standard MFCC and PLP features. Kaldi support most commonly used feature extraction approaches: e.g. VTLN, cepstral mean and variance normalization, LDA, STC/MLLT, HLDA, and so on.

2.1.2 Acoustic Modelling

An acoustic model is used in Automatic Speech Recognition to represent the relationship between an audio signal and the phonemes or other linguistic units that make up speech. The model is learned from a set of audio recordings and their corresponding transcripts. It is created by taking audio recordings of speech, and their text transcriptions, and using software to create statistical representations of the sounds that make up each word.modern speech recognition systems operate on the audio in small chunks known as frames with an approximate duration of 10ms per frame. The raw audio signal from each frame can be transformed by applying the mel-frequency cepstrum. The coefficients from this transformation are commonly known as mel frequency cepstral coefficients (MFCC)s and are used as an input to the acoustic model along with other features.Kaldi supports conventional models (i.e. diagonal GMMs) and Subspace Gaussian Mixture Models (SGMMs).

Gaussian Mixture Models

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians. Kaldi support GMMs with diagonal and full covariance structures. Rather than representing individual Gaussian densities separately, it directly implements a GMM class that is parameterized by the natural parameters, i.e. means times inverse covariance and inverse covariance. The GMM classes also store the constant term in likelihood computation, which consist of all the terms that do not depend on the data vector. Such an implementation is suitable for efficient log-likelihood computation with simple dot-products.

Speaker Adaptation

Kaldi support both model-space adaptation using maximum likelihood linear regression (MLLR) and feature-space adaptation using feature-space MLLR (fMLLR), also known as constrained MLLR. For both MLLR and fMLLR, multiple transforms can be estimated using a regression tree. When a single fMLLR transform is needed, it can be used as an additional processing step in the feature pipeline. The toolkit also supports speaker normalization using a linear approximation to VTLN, similar to, or conventional feature-level VTLN, or a more generic approach for gender normalization which we call the exponential transform. Both fMLLR and VTLN can be used for speaker adaptive training (SAT) of the acoustic models.In ths project we used MLLR for speaker adaptation.

2.1.3 Training

All kaldi training and decoding algorithms use Weighted Finite State Transducers (WFSTs). In the conventional recipe, the input symbols on the decoding graph correspond to context-dependent states (in our toolkit, these symbols are numeric and we call them pdf-ids). However, because it allows different phones to share the same pdf-ids, we would have a number of problems with this approach, including not being able to determine the FSTs, and not having sufficient information from the Viterbi path through an FST to work out the phone sequence or to train the transition probabilities. In order to fix these problems, kaldi put on the input of the FSTs a slightly more fine-grained integer identifier that we call a transition-id, that encodes the pdf-id, the phone it is a member of, and the arc (transition) within the topology specification for that phone. There is a one-to-one mapping between the transition-ids and the transition-probability parameters in the model: we decided make transitions as fine-grained as it could without increasing the size of the decoding graph.

Kaldi decoding-graph construction process is based on the recipe described in [14]; however, there are a number of differences. One important one relates to the way we handle weight-pushing, which is the operation that is supposed to ensure that the FST is stochastic. Stochastic means that the weights in the FST sum to one in the appropriate sense, for each state (like a properly normalized HMM). Weight pushing may fail or may lead to bad pruning behaviour if the FST representing the grammar or language model (G) is not stochastic, e.g. for back off language models. Kaldis approach is to avoid weight-pushing altogether, but to ensure that each stage of graph creation preserves stochasticity in an appropriate sense. Informally, what this means is that the non sum- to-one-ness (the failure to sum to one) will never get worse than what was originally present in G.fst.

2.1.4 Decoding

Kaldi has several decoders, from simple to highly optimized; more will be added to handle things like on-the-fly language model rescoring and lattice generation. By decoder we mean a C++ class that implements the core decoding algorithm. The decoders do not require a particular type of acoustic model: they need an object satisfying a very simple interface with a function that provides some kind of acoustic model score for a particular (input-symbol and frame)

2.1.5 Language Modelling

A statistical language model is a probability distribution over sequences of words. Given such a sequence, say of length m, it assigns a probability P(W1,....,Wm) to the whole sequence. Having a way to estimate the relative likelihood of different phrases is useful in many natural language processing applications, especially ones that generate text as an output.

Since Kaldi uses an FST-based framework, it is possible, in principle, to use any language model that can be represented as an FST. Kaldi provide tools for converting LMs in the standard ARPA format to FSTs. In its recipes, kaldi have used the IRSTLM toolkit 3 for purposes like LM pruning. For building LMs from raw text, users may use the IRSTLM toolkit, for which we provide installation help, or a more fully-featured toolkit such as SRILM.

2.1.6 Finite State Transducers (FST)

Much of current large-vocabulary speech recognition is based on models such as HMMs, lexicons, or n-gram statistical language models that can be represented by weighted finite-state transducers.

A FST is a finite automaton2 whose state transitions are labeled with both input and output symbols. Therefore, a path through the transducer encodes a mapping from an input symbol sequence, or string, to an output string. A weighted transducer puts weights on transitions in addition to the input and output symbols. Weighted transducers are thus a natural choice to represent the probabilistic finite state models prevalent in speech processing

2.2 Generic Acoustic Model

We need a generic Acoustic model built on training data for a good working model of kaldi based software.Initially MFCC features are extracted and CMVN for the training data set.We have three GMM-HMM training and decoding stages

- MonoPhone Training & Decoding

- tri1 : Deltas + Delta-Deltas Training and Decoding

- tri2 : LDA + MLLT Training and Decoding

During each step we first train the data and then makegrap for decoding and at last decoding is done.

Chapter 3

Adaptive Long Speech-Text Alignment

We implemented the adaptive, iterative speech-text alignment algorithm. As mentioned earlier, at the core of the algorithm lies the assumption that the long speech-text alignment problem can be posed as a long text-long text alignment problem given a well performing speech-text conversion tool, i.e., speech recognition engine. Given that the text-text alignment problem can usually be solved quite efficiently even for long text using dynamic programming to minimize the Levenshtein distance between the reference and the hypothesized text, the main bottleneck is then at the speech recognition part.

Now we have the generic acoustic model built on training data set we need a systematic algorithm which implements the Long speech-text alignment. Sailalign has designed a standard algorithm we made some changes and implemented in Kaldi. The main difference between Sailalign and our software is Sailalign uses HTK models for speech recognition and acoustic modelling we have used GMM-HMM models which produces better results.

3.1 Algorithm

3.1.1 Initialization

Initially we take the time aligned transcription, then find teh acoustic features of test date set. We now apply speech regions by using Voice Activity Detection(VAD). The audio stream has to be segmented into smaller chunks whose duration is constrained by computational limitations of the speech recognition engine used (approximately 10 to 15 seconds in our case). To avoid cutting a word into two, segmentation is guided by a voice activity detection module. For efficiency, segmentation is performed in the acoustic feature domain and not in the audio domain. In this way, acoustic feature extraction is carried through only once and not repeated for every repartitioning of the input stream. Repartitioning will be necessary at subsequent iterations. Acoustic features are extracted from the audio stream. To ensure that the speech recognition output will be as close as possible to the reference transcription, a transcription specific language model is built at this point.

3.1.2 Speech Recognition, Text-Text Alignment

Continuous speech recognition is then applied to identify the lexical content of the individual speech segments. The hypothesized transcripts are concatenated into a single one, which is then aligned with the reference transcript. Reliably aligned regions are selected by applying a minimum-number-of-words criterion, i.e., they should include at least a minimum number of consecutive aligned words. The rest of the audio is considered to be unaligned and it is repartitioned into segments of appropriate length. The transcription is also partitioned appropriately this time to leave the aligned regions out. The recognition-text alignment cycle will be repeated for only the unaligned audio and text segments.

3.1.3 Acoustic and Language Model Adaptation

To improve noise robustness, we adapt the acoustic models at each iteration in a supervised manner using the reliably aligned regions. Maximum Likelihood Linear Regression is applied and adaptation is performed in two steps. First, we train a global transformation and then, for groups of phonemes in which we have sufficient adaptation data, we build a class-based transformation. The language models are also updated so that they are trained specifically for each unaligned region. This process, i.e., recognition-alignment-adaptation, is iterated three times. In the subsequent two iterations, the acoustic models are not adapted, and the language model is described by a constrained finite state grammar which only allows the expected sequence of words for the segment (and insertions/deletions for the fourth iteration). This is expected to further increase the number of aligned regions in the case of very noisy audio.

3.1.4 Psuedo Algorithm :

1:Time-aligned transcription (S, T)

2:Detect speech regions by Voice Activity Detection (VAD)

3:Extract acoustic features A from the audio signal

4:$E_0 <-$ Generic acoustic models

5:$U_0 <-$ (A,S) Unaligned acoustic features and the corresponding word sequence

*6:**for** i=1 to 5 **do***

*7: **for all** N segments in U_{i-1} **do***

8: $A_n <-$ acoustic features of the segment

9: $S_n <-$ corresponding word transcript

10: Segment A_n in K_n subregions A_{nk} of approximate duration D

11: ***if*** *i* < 4 ***then***

12: *Build a trigram language model* L_n *on* S_n

13: ***else***

14: *Build a finite state grammar* L_n *on* S_n

15: ***if*** *i = 5* ***then***

16: *Do not allow insertions or deletions*

17: ***end if***

18: ***end if***

19: ***for*** *k=1 to* K_n ***do***

20: (R_{nk}, T_{nk}) *= SpeechRecognition(*A_{nk} *,* E_{i-1}*,*L_n*)*

21: ***end for***

22: ***end for***

23: *(R, F)* $<-$ $S_{n,k}(R_{nk}, T_{nk})$

24: *Use Dynamic Programming to minimize Levenshtein distance*

25: ***if*** *i ¡ 4* ***then***

26: $E_i <-$ *Adapted acoustic models using regression class tree based (****MLLR****)*

27: ***esle***

28: $E_i <- E_3$

29: ***end if***

*30:****end for***

3.2 Results

Here the first column contains a hindi word and following two columns corresponds to starting and end time of occurrence of that particular word in audio file

Chapter 4

Documentation

This chapter contains all the work we have done and the changes we have made on any standard codes.It also acts as a documentation for the people who want to carry our further Whole project has been uploaded in GPU's present in lab under the user name Batch5/dileep/longaudio.

4.1 Data set Locations

As mentioned above we have taken modi's mann ki baat monthly speeches as our data set. The chunked train data is present in data/train/training_data. Similarly test data in data/test/testing_data

Since we did not have time left before presentation there are some extra audio files which can be used for training data which are present in data_mann_ke_bhat/training data

Note: If you are adding new data make sure that they are punctuations and whole language be unanimous, remember only chunked file of data around 1 minute can be stored in training data set present in data/train/testing_data, for test we can directly stored in data/test/testing_data

4.2 lexicon building

As mentioned earlier we have used unified software for lexicon building. It also has some other regional Indian languages which you can work on.

Files we build using unified

- **lexicon.txt** - it contains all the different words and there phones
- **nonsilence_phones.txt** - All silence phones
- **optional_silence.txt**
- **phones.txt** - All phones including silence phone
- **silence_phones.txt** - Silence phone(sil)
- **text.txt** - Complete text of training data

Since we did not have prior dictionary we have used only the words which appeared in data set as dictionary (note - we also have added the test data set to dictionary).

All the files are stored in data/local/dict.

Run the python file data/train/train_python.py to create files wav.scp,utt2spk,spk2utt,text,corpus

Language Model

Initially for building a generic acoustic model on training data we need a language model this generates files which are helpful in building acoustic model

- **phone_lm.sh** - Build language model on training data

4.3 Acoustic model

For feature extraction and building acoustic model run 22run.sh file

Adjust the Acoustic model parameters according to the data set present we have tried with different values and fitted with most suitable values.

There are three stages in model building. Initially MFCC features are extracted and CMVN are calculated.

steps/$make\_mfcc.sh$ *–cmd* "$train\_cmd$" *–nj 20 data/*$xexp$*/make_mfcc/*$xmfccdir$

In the above line number 20 represents no of parallel jobs done so we change this no according to system available to increase computational speed.

Stages of GMM-HMM

- **MonoPhone Training & Decoding**
- **tri1 : Deltas + Delta-Deltas Training & Decoding**
- **tri2 : LDA + MLLT Training & Decoding**

Now there three GMM-HMM stage and for each stage we have training, make graph and decoding.

First run the training step in each stage of GMM-HMM but commenting the other two steps,then makegraph and at last decoding.

Adjust the values of sen and gauss in tri1 and tri2 according to training data set available.

Make sure that in cmd.sh we set train_cmd = queue_cpu.pl and decode_cmd = queue_cpu.pl if you are running on GPU else set both to run.pl(if you are running on your own system).

4.4 Adaption Long Speech-text Alignment

First make sure all the Kaldi files which are used are included in path.sh, we have tried to include most of the files you can edit if some are missing.

4.5 longaudio_multi_dir.sh

Running this file enough to complete the required job.

In line 15 make sure that the file selected is same as the file present in test data set.

In the line 18 initially run by keeping value of stage 1 then change it to 2.

4.5.1 longaudio_alignment.sh

Here the complete algorithm is implemented

- **make-feats.sh** - mfcc and cmvn are calculated.
- **compute-vad** - This a in build c++ function used for voice activity detection and stores the results in vad.ark.
- **split_vad.py** - Splitting of vad.ark files into small chunks of 10 seconds.
- **sym2int.py** - Preparation of text files of 10 seconds length in audio file.
- **build-trigram.sh** - For preparing trigram LM.
- **build-graph-decode-hyp.sh** - For building makegraph and decoding.
- **make-status-and-word-timings.sh** - This is main file which calculates word timing of each word in the text.

Now after this there a continues for loop which runs 5 iterations building a better models for each iterations.

4.6 Further Works

4.6.1 MFCC

Since we have used single speaker in all the audio file we used so there is no must noise rubustness present but if you want to adapt to a new user they implementation of MFCC is necessary.

4.6.2 Adding of Extra data to Training Data set

Though we had 9 hours of data but we were able to just chunk 3 hours of data and we build a model on just 3 hours of data. The further work can be once we get the results of test data we can chunk the audio file of approx 1 minute and add to train data set, we can continue this for all the extra files.

Chapter 5

Bibliography

[1] P. J.Moreno and C. Alberti, A factor automaton approach for the forced alignment of long speech recordings, in Proc. IEEE Intl Conf. Acous., Speech, and Signal Processing, 2009.

[2] D. Caseiro, H. Meinedo, A. Serralheiro, I. Trancoso, and J. Neto, Spoken book alignment using WFSTs, in HLT02 Proceedings of the second international conference on Human Language Technology Research, 2002.

[3] J. Yuan and M. Liberman, Vowel acoustic space in continuous speech: An example of using audio books for research. in CatCod, 2008.

[4] A. Ljolje and M. Riley, Automatic segmentation and labeling of speech, in Proc. IEEE Intl Conf. Acous., Speech, and Signal Processing, 1991. [5] G. Margolin, P. H. Oliver, E. B. Gordis, H. G. OHearn, A. M. Medina, C. M. Ghosh, and L. Morland, The nuts and bolts of behavioral observation of marital and family interaction, Clinical Child and Family Psychology Review, vol. 1, no. 4, pp. 195213, 1998.

[6] J. Gottman, H. Markman, and C. Notarius, The topography of marital conflict: A sequential analysis of verbal and nonverbal behavior, Journal of Marriage and the Family, vol. 39, no. 3, pp. 461477, 1977.

[7] J. Jones and A. Christensen, Couples interaction study: Social support interaction rating system, University of California, Los Angeles, 1998. [Online]. Available: http://christensenresearch.psych.ucla.edu/

[8]M. P. Black, A. Katsamanis, C.-C. Lee, A. C. Lammert, B. R. Baucom, A. Christensen, P. G. Georgiou, and S. Narayanan, Automatic classification of married couples behavior using audio features, in Proc. Intl Conf. on Speech Communication and Technology, 2010.

[9] P. Moreno, C. Joerg, J.-M. van Thong, and O. Glickman, A recursive algorithm for the forced alignment of very long audio segments, in Proc. Intl Conf. on Spoken Language Processing, 1998.

[10] C. J. Leggetter and P. Woodland, Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models, Computer Speech and Language, vol. 9, pp. 171185, 1995.