Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Two-Stage Utterance Detector for Speech Recognition

Nir Galili

Term Project SA-2009-08

Institut für Technische Informatik
und Kommunikationsnetze

Supervisor:

Dr. B. Pfister and M. Gerber

Responsible Professor:

Prof. Dr. L. Thiele

29 May 2009

Declaration

I affirm that I have read the information notice on plagiarism,[1] independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Place, Date

Signature

____

Name Surname

Note

The above declation is a mandatory part of the report of each semester and master thesis at TIK. In case of multiple authors, the text of the declaration has to be adapted accordingly and signed by each author.

[1]Information notice by the Rector of ETH Zurich http://www.ethz.ch/faculty/education/plagiarism_l_de.pdf

Contents

Summary

Recognizing isolated words is a variant of Automatic Speech Recognition (ASR). Examples of applications that need such functionality are mobile phone speech activated speed dialing and voice controlled machine. A necessary preliminary step for such a task is to detect the borders of separated utterances, i.e. distinguish between silence or noise part and speech sequence, from which information later is to be extracted.

In common applications, signal energy is an important factor in detecting the presence of an informative utterance. However relying on signal energy alone would lead to non-optimal performance.

The goal of this project is to find a method to refine signal energy detection of utterance borders, using additional speech features. The method includes Hidden Markov Model with speech, silence and specific non-speech classes. All states are modeled with Gaussian Mixtures Models. Sound features that are used are Mel Frequency Cepstral Coefficients.

1 Introduction

1.1 Project Scope

A block diagram of a general word recognition module is given in figure 1. The sound data input is processed by the feature extraction module. Commonly used features are Mel Frequency Cepstrum Coefficients (MFCC). The word recognizer module estimate state path of the utterance input data, compares it with known paths of vocabulary words, that had been found during a training process, and choose the word with the closest path.

The utterance detector module estimates which part of input data is the relevant utterance, farther to be processed by the word recognizer module. The project deals with improving utterance detector module, which is an important preprocessing stage in the word recognition scheme.

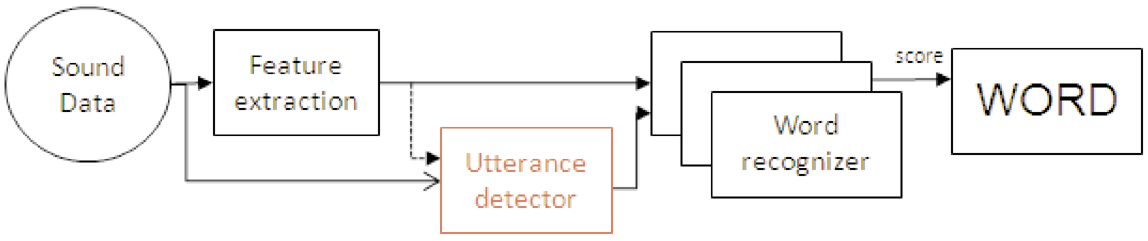

Figure 1: *Word recognition general box diagram*

1.2 Problem

The utterance detector used currently at the institute detects the utterance boundaries only based on the signal energy. In common applications, the input microphone is usually close to the speaker, and therefore speech signal power is considerably bigger than noise power. This is the reason that signal energy is a good indicator for utterance presence. Relying on signal energy alone, however, occasionally is not enough. High energy noise for example, might cause utterance false detection, while low energy speech signal such as some unvoiced sounds could be mistakenly considered as non-speech. The goal of the project is to improve energy utterance detector. We would like to minimize the complexity of the new designed detector. Using 'standard' components that are commonly used in other stages of word detection pipeline is preferable. It is expected that using additional, more informative sound features beside signal energy, would lead to an improvement in the detector performance.

2 Implementation

2.1 Modeling data with Hidden Markov Model (HMM)

Sound data could be modeled as generated by a state machine. Different states correspond to different classes of sound. Two necessary such classes are noise class and speech class. In addition specific non-speech classes, like music class, could be added. Utterance starting delimiter is decided upon at the frame in which speech state is entered, and terminating delimiter is placed at a frame in which the machine exits speech state. An initial design of such a model is given in figure 2

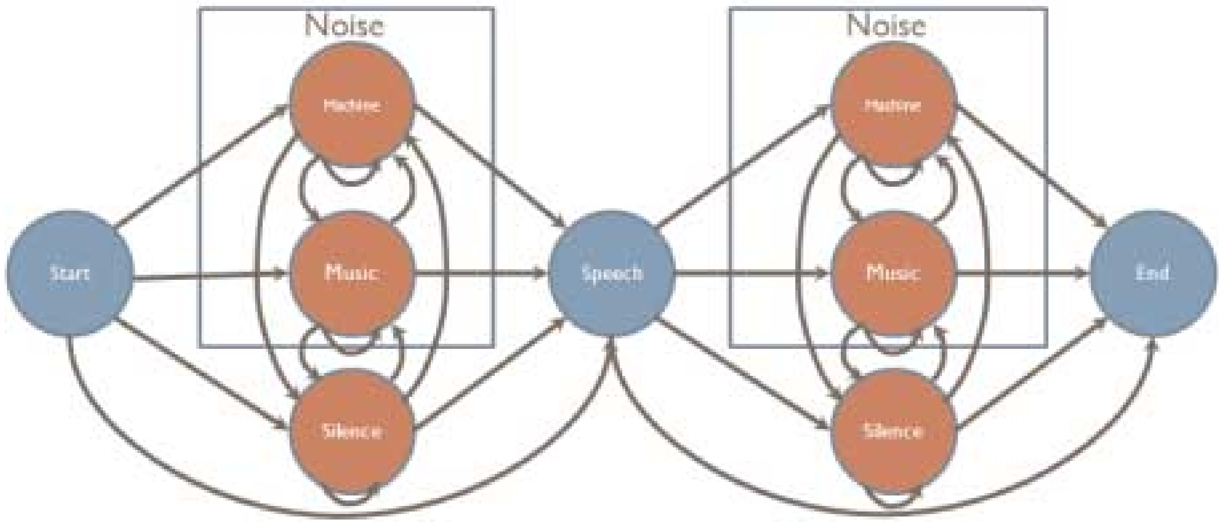

Figure 2: *HMM models transition between different classes of sound.*

The probability of transitions between classes are chosen to be equal, so there is no prior assumption on the dynamics of moving from one state to another. The individual state probability distribution functions (PDF) are assumed to be mixtures of Gaussians (using GMM - Gaussian Mixture Model). The Gaussians means and variances, as well as the relative weight of each Gaussian in the mixture, are parameters of each specific state. Modeling the HMM states is done with Mel Frequency Cepstral Coefficients (MFCC) as the sound data features. Mel Frequency Cepstrum is a representation of the short term power spectrum of a sound, based on a Discrete Cosine Transform of a log power spectrum on a non linear mel scale of frequency[3]. Additional information about HMM can be found at [1],[2].

2.2 Training Method

The state specific Gaussian mixtures parameters values are derived during a training process. The input for such training are class specific groups of signals. The following 3 state machine is built for the training task (figure 3)

Note that the above 3 state machine is used only to estimate the Gaussian mixtures param-

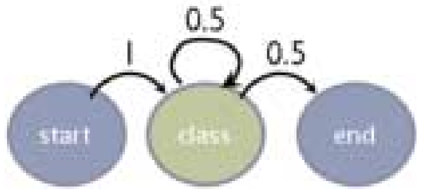

Figure 3: *3 state HMM machine, used for trainig the model.*

eters which characterize the class specific states in the general HMM model (figure 2). We are not interested in the transition probabilities of the 3 states machine derived from the training process.

2.3 Starting point

The starting point of the project was given code for training HMM machines (including estimation of optimal state path) and extracting data features module. The first task was to understand the HMM training code, and the training results data structure (cdhmm variable). A function was written to visualize the Gaussian mixture models of the different classes. It produces a 3D plot of random points that are distributed according to class parameters. I used principle component analysis to present the most separable dimensions. In addition the function can plot 2D histogram of these random points.

2.4 Training Parameters

The utterance detector model was trained on speech, silence and sewing noise class sequences. Mfcc probability distribution is composed of up to 64 Gaussian mixtures. (Initial trial was to train up to 256 Gaussian mixtures, but the training program failed to train the specific silence class data - values of logZeta variable were too small and lead to sumZeta be zero, as a result of division by zero some of the mixtures variables got ’NaN’ values . Debugging the problem was time consuming and is out of the project scope)

The number of Gaussian mixtures that is used is a tradeoff. High number of mixtures allows flexibility to adjust and characterize the big variety of speech class data and noise class data. On the other hand, the time needed to train the model and model complexity grows with the number of Gaussian mixtures used.

2.5 Modifying HMM model

When testing the first proposed general HMM model above (figure2), It could be noticed that when there where speech signal overlapping with non-speech class data, the utterance delimiters tended to be set inside the speech part, i.e. the start or end parts of speech area were marked as non speech. This caused undesirable degradation in performance. Analyzing the different classes probability for overlapping class data (figure 4), it can be seen that when speech and non-speech data overlap, the probabilities of the classes "compete". If there are dominant non-speech classes parts at the edge of the speech, near the transition to or from speech sequence, the current configuration of the HMM state machine would prefer to append these parts to the non-speech states. To deal with this issue the general HMM machine scheme was revised to the one depicted in figure 5

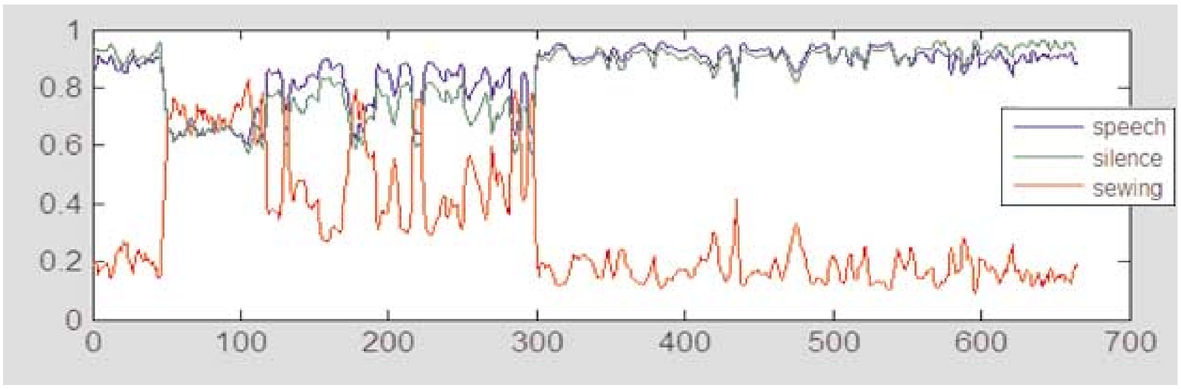

Figure 4: *Class probabilities graph. For each data window class probabilities were calculated. The above graph is the result of concatenating probabilities from consequent data frames.*

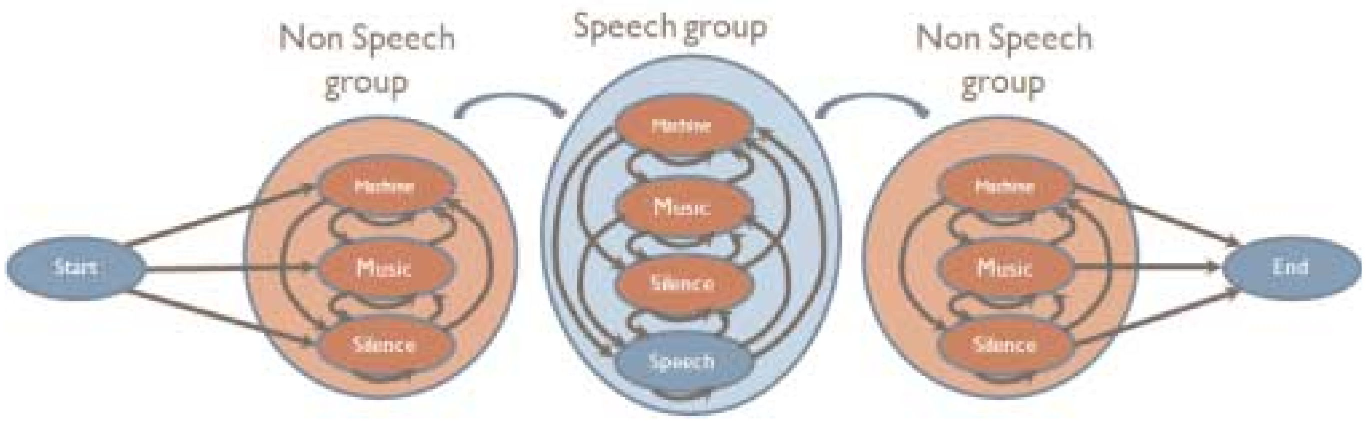

Figure 5: *New proposed HMM diagram.*

Instead of separate speech state, the new HMM configuration contains a "group of speech states" that includes both speech and non speech states. The states in the group can alternate, similar to the way the machine can move freely among the states at the "non-speech groups". Transition between each group to another is allowed from any state in one group to any other state in the next group (group that is closer to the end state). However transition to a previous group is not allowed. The utterance start is at the transition into the speech group, and the utterance end is on the move from that group. An example of the described behavior can be seen in figure 6

The new model can tolerate competition of classes during utterance sequence (speech group sequence). A new problem that might emerge is that the machine would tend to stay at the speech group through longer periods than necessary. This would lead to deciding upon too large utterance sequences. The solution is to set prior that would cause being at the non-speech group of states more desirable. That way the staying in the speech group would be worthwhile only in case that there are enough high probability speech sequences - the right motivation to decide about utterance sequence. The prior was set by multiplying the Gaussian mixtures weight of the speech state group by a factor smaller than 1.

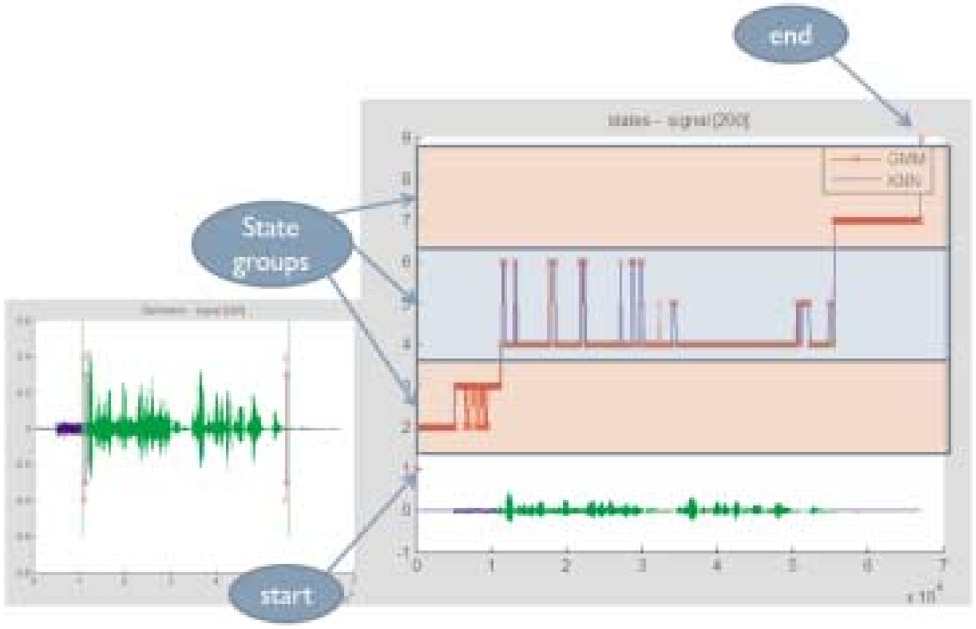

Figure 6: *Right: Group states dynamics sample. The light red blocks are corresponding to the non - speech state groups while the light blue block mark the speech group states. On the bottom of the graph given the sound data signal as reference. The utterance part is green. The rest of the signal is blue. Left: the corresponding utterance decision.*

2.6 KNN

An alternative way to get class probabilities estimation of a new data is to use K Nearest Neighbor (KNN) method. The idea is to modify the function that calculates path probabilities such that it would use states probabilities based on K Nearest Neighbor instead of using the matlab function 'normpdf' to calculate these probabilities based on state Gaussian mixtures parameters. The KNN state probabilities are calculated using auxiliary KNN functions: one function is used to get features and correspondent classes from training data, another function, given new data features, calculates their class probabilities. These are later plugged into the function that calculates optimal path of the HMM.

3 Tests

3.1 Training

The model was trained on 3 classes: speech, silence and sewing noise. There were available about 15,000 files of speech class training, 5000 files for silence class training and three long sequences of sewing machine noise with different sewing speeds (one sequence with changing sewing speed) for the third class training. Two test sets were done: In the first test set,the number of Gaussian mixtures used in the models was changing, while the training data kept constant (All training files were used). On the second test set, the number of files used for the training changed (speech and silence classes), while the models had same number of Gaussian mixtures (32 mixtures). The different models in each test set were tested on 258 test signals. The test signals were composed of silence sequence followed by speech sequence and a second ending silence sequence. In addition Sewing noise was added such that it overlapped the end of first silence sequence and the start of the first speech sequence (figure 7)

3.2 Test Framework

Test framework was built in a Matlab script. Its purpose is to test the HMM model (figure 5) on the test data files. The framework includes building the HMM model from training result structures as a preprocessing stage. This allows using different training result parameters in the testing framework. A post processing stage allows debugging a specific signal, plotting intermediate results of its processing.

The script first loads training data from 3-state machine trainings, then builds the main HMM machine using training parameters. Building the HMM, the transition probabilities of each state inside a group are set to be equal. On each test file feature data extraction is done. An auxiliary function finds the most likely state path corresponding to the input data. The indices of frames which are within the speech group states are extracted into a vector. The first element of the vector is the estimated start of the utterance frame, and the last element is the utterance estimated end frame. Utterance delimiters results are kept in a cell structure. For the error estimation, average absolute difference between the estimated utterance border and the hand marked ground truth utterance borders is calculated. There exists the possibility to debug a chosen test file. Test results that can be viewed are: estimation of utterance boundaries, optimal state path, and frames class probabilities.

4 Results

All tests summed in table 3 and table 2 show improvement over the reference energy detector (table 1). As expected, using other sound data features in addition to energy improves the utterance detection.

However in contrast to the expected, training data with a greater number of Gaussian mixtures resulted in a bigger average error. This might be due to over fitting of the model to the training data. When training with more mixtures the probability distribution functions tended to

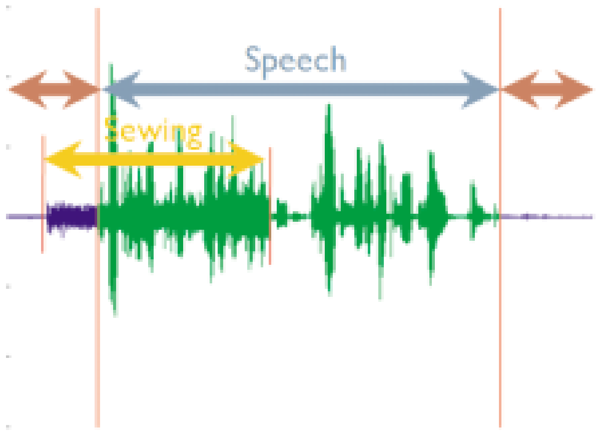

Figure 7: *Scheme of test signals*

	Energy Detector
Error [abs(Samples offset) x 10^3]	5.635

Table 1: *Energy detector, reffernce.*

have smaller variance.

On the second test set, models with different number of training files were tested. It can be seen that with around 600 and more training files the performance error is almost constant. It was expected that training the model with as few as 10 training files, will result in a bigger error than other tests, however, as before, it might be that on test with the bigger error we suffer from over fitting of the model to the training data.

5 Further work

5.1 Performance Tests

As can be concluded from the results section, it would be valuable to further test the algorithm with different parameters configuration among them different number of Gaussian mixtures, and varying training data set size. In addition it would be favorable to test model with a bigger testing set.

5.2 Assigning weights to mfcc coefficients

Currently, all data feature coefficients are treated in the same manner, despite the fact that their contribution to the classification is different. The first mfcc coefficient, which is related to signal energy, probably has a more important role than other features. In fact the current utterance detector performs generally well relaying on the energy of the signal only. An automatic method

Number of mixtures	1	2	4	8	16	32	64
Error [abs(Samples offset) x 10^3]	0.995	1.856	2.572	3.160	3.541	3.843	3.886

Table 2: *Tests results for models with different number of mixtures, same training data (all trainig files were used)*

Number of Training Files	10	600	1500	All Files
Error [abs(Samples offset) x 10^3]	1.058	4.066	3.923	3.843

Table 3: *Tests results for 32 mixtures models, with different amount of training data*

to estimate each data feature contribution to the classification, and weighting these coefficients accordingly could improve system performance. A way to measure feature contribution could be for example detecting feature false classification rate. The more misclassification a feature has, the less weight it should have in decisions.

5.3 Additional data features

Only mfcc data features were used in training and estimating data class probabilities. Integrating other kinds of data features could be beneficial. Such features could be Zero Crossing Rate[4], and the difference of the mfcc features between consequent frames.

References

[1] *Foundations of Statistical Natural Language Processing - Christopher D.Manning, Hinrich Schutze.*

[2] *Statistical Methods For Speech Recognition - Jelinek.*

[3] *Wikipedia, Mel-Frequency Cepstrum.*

[4] *Min Yang et al, Enhance Speaker Segmentation by Elaborating Utterance Detection.*

A Code description

A.1 Test Framework Functions

- checkStatistics.m For each training configuration, the script builds hmm from training configuration data. It loops on the test files. For each file sewing machine noise is added and mfcc data features are extracted. The script uses the function log_cont_viterbi_alg.m to find optimal states path. If TEST_KNN flag is on, it uses functions that would be describe later to find optimal path for KNN class probabilities estimation. Frames that are within the speech group states are found and are put into a vector, then the scripts sets utterance start delimiter as the first frame, and the end delimiter as the last frame, and calculates the difference to the hand marked ground truth utterance borders.

  Debug: given signal number, the script plots the signal with the estimated and the ground truth utterance borders. It uses log_cont_viterbi_alg to calculate optimal state path, and plots it. It uses check_cdhmm_general.m function and the training configuration to calculate frames class probabilities, and to plot them.

- addSew.m adding sewing noise to test files.

A.2 KNN functions

- train_KNN.m extracts mfcc features from different classes data. Saves the features at ’trainingData’ variable and correspondent class index in ’trainingClasses’ variable

- KNN_prob(data,trainingData,trainingClasses,K) Input : data features, ’trainingData’ and corresponding ’trainingClasses’ variables that are the output of the train_KNN function. K - number of nearest neighbors Output: probabilities of data

- B = CreateBeta(cep,trainingData,trainingClasses,K,amp), Uses KNN_prob function to calculate data Class probabilities, makes B matrix that would be used later in function log_cont_viterbi_alg_KNNbased

- log_cont_viterbi_alg_KNNbased.m modified log_cont_viterbi_alg function that uses Beta calculated in CreateBeta instead of calculating it with the function normpdf and Gaussian mixtures parameters.

A.3 Auxiliary File processing functions

- FPreproc.m. pre-processing wav files and delimiter data files. Script first check insures match between data ’wav’ files and delimiters data ’lab’ files, then it cuts the wav files into smaller class specific files and save files of each class in a separated folder. Later these class specific files are used for training the model. In Addition the script can save into a variable the delimiters of utterance manually marked (ground truth)

- PlotBorders.m plots delimiters.

- choose_random_files.m chooses files from a big training set , skips files so the chosen training representatives would be diverse.
- run_v2detector.m v2_detector functions wrap detect_utterance_v2, and an implementation of the energy detector that is compared to the hmm detector. Function runs it on signals from a given path and saves the delimiters resulted in a cell structure.