Internal Use Only（非公開）

TR－SLT－0001

Start-Endpoint-Detection for SPINE2 Using Energy- and GMM-based Methods

Norbert Binder Rainer Gruhn Konstantin Markov

December 18th, 2001

This report describes the start-endpoint-detection module (SED) used at ATR-SLT for the SPINE2 evaluation. For the speech/non-speech separation a combination of the energy-based EPD-system (used in SPINE1) and a GMM-based method was realized.

（株）国際電気通信基礎技術研究所
音声言語コミュニケーション研究所
〒619－0288 京都府相楽郡精華町光台二丁目 2 番地 2 TEL：0774－95－1301

Advanced Telecommunication Research Institute International
Spoken Language Translation Research Laboratories
2-2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-0288,Japan
Telephone:+81-774-95-1301
Fax :+81-774-95-1308

Contents

1 Introduction **4**

2 Software **5**
2.1 CB Training 5
2.2 GMM Training 5
2.3 GMM Testing 6
2.4 HTK Toolkit 6

3 Task: SPINE evaluation **6**
3.1 Database 6
3.2 Used features 7
3.3 Likelihood-Based Measure 7
3.4 Using Single Frames 7
3.5 Using n Frames 7
3.6 Implementation 8
3.7 Realization of The Single Frames Method 8
3.8 Realization of The Multi-Frame Method 8
3.9 Energy Based Endpoint Detection 9
3.10 Combination Method 9

4 Estimating The Threshold **10**

5 Method Comparison **11**

A Software **13**
A.1 Structure 13
A.2 extract.py 14
A.3 Feature Extraction With HTK 14
A.4 noheader.py 14
A.5 mfcc-check.py 14
A.6 datgen.py 15
A.7 frame2time.py 15
A.8 dataprep.py 15
A.9 histo_gen.py 16
A.10 CB- and GMM-training 16
(A.10.1) TrainCB 16
(A.10.2) TrainGMM 16
(A.10.3) Histo 17

B Evaluation Results **18**

References **19**

1 Introduction

This report introduces an approach to separate speech and non-speech using a speech- and a noise-trained GMM. For each GMM the log-likelihood for each GMM and then their difference is calculated and compared to a threshold.

Additionaly a combination of the existing energy-based EPD system, which was used for SPINE1 evaluation [MMG+01], and the GMM-based method is introduced. It could improve the GMM-based system especially for noisy environments.

The main section will give an introduction to the method itself. In section A different software modules and their usage in the separation-system are explained. A comparism of the results for different methods (GMM, EPD, combinatin) and settings is given in the appendix.

2 Software

The project was supposed to base on the given software from Konstantin Markov an invited researcher at ATR-SLT, which contained one program each for:

- Codebook (CB) Training
- Gaussian Mixture Model (GMM) Training
- GMM Test

2.1 CB Training

The program calculates a codebook from a training set of feature vectors. This is done by combining similar feature vectors to clusters. For each cluster a center vector the so called centroid is calculated which is the representant of this cluster. The number of centroids is limited to a certain number. Each time a new feature vector is added the distances to all centroids or - depending on the method to the n-next neighbours are calculated and the vector is then added to the nearest cluster. The centroid of this cluster is has to be recalculated before the next vector can be added. This way all feature vectors of a training set are added. Finally the centroids are saved as Codebook.

2.2 GMM Training

The GMM Training software trains a *hidden Markov Model (HMM)* by using the same testset like the codebook training and the there calculated codebook. A HMM is defined by the number of states, the transition probabilities between this states and the distribution of the output probabilities from each each state. Another limitation on the HMMs is done as the GMM can only be entered at the first state and be left at the last state and the number of transitions is limited. Here the distributions are modeled by superpositioning *Gaussian distributions*. Thus this HMMs are called *Gaussian Mixture HMM* or short *GMM*.

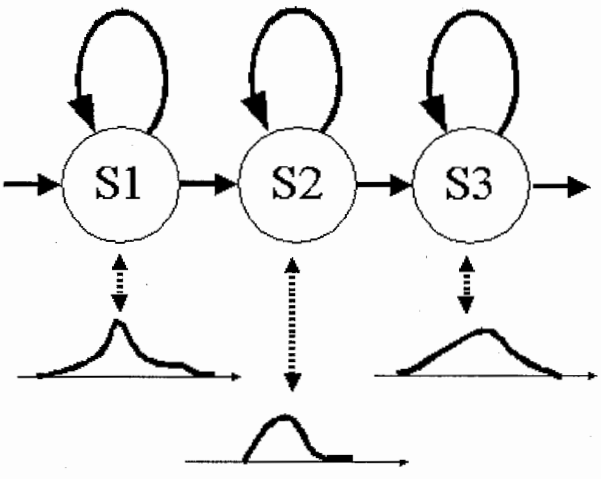

Figure 1: A HMM with three states and the output probabilities for each of these states

2.3 GMM Testing

With the third program the calculated GMM can be tested. It calculates the probability that the sequence of feature vectors given in the test file could be emitted with the given GMM.

2.4 HTK Toolkit

The HTK toolkit offers a set of tools necessary for speech processing. This includes the calculation of different feature coefficients like linear prediction coefficients (LPC) or MEL frequency cepstral coefficiants (MFCC) which I used for my work. Other tools calculate hidden Markov models (HMM) or help creating acoustic models (AM). The number of tools and functions is too high to be discussed here in depth.

3 Task: SPINE evaluation

The task of this project is to create a system for speech and non-speech separation based on GMMs [BMGN01]. The created library function should be used for the SPINE 2 evaluation which tests speech recognition in noisy military environment. Due to the fact that a lot of different noises with different characteristics are used, the idea of using acoustic models for the separation seemed to be more promising than the usage of an energy based which was included in the SPINE1 recognizer (see section 3.9). CMU, another participator at this evaluation achieved very good results with a comparable method [SSRS01]. Their system and the results were published in the paper *Speech in Noisy Environments: Robust Automatic Segmentation, Feature Extraction and Hypothesis combi nation* at *ICASSP 2001.*

3.1 Database

The data I was using for testing and evaluation was the SPINE1 corpus (see also section 3). It consists of stereo data of conversations of 20 speaker pairs. Each speaker talks on one channel and has a different background noise from now on also called environment. The B channel has a silent or nearly silent office noise background while the A channel contains one of the following noise types:

dd, quiet	quiet	helo	helicopter
office	nearly quiet	bradley	tank
ar, HMMWV	Army	f16	F16-fighterjet
af, AWACS	Air Force (E3A AWACS)	car	inside-car noise
nv, navy	Aircraft Carrier CIC	street	street noise

The SNR for this environment varied from 23dB for the dd-type and 16dB for the ar-type to below 15dB for the nv environment.

From each speaker pair fourteen conversations with a duration of 5 minutes were recorded (about 720 minutes total). The test set for the evaluation contains 10 hours of conversation from 40 Speakers (20 speaker pairs) in 6 noise environments thus 20 additional unknown speakers and two unknown environments are included.

3.2 Used features

After the analog-digital conversion of the speech data the amount of information is too high for processing. Thus those features have to be extracted which are important for the future processing. This can be differ from task to task. Common features are for example LPC or MFCC coefficients, the number of zero-y-axis crossing or informations about the phase of the signal. Typically MFCCs are used in modern speech recognition.

3.3 Likelihood-Based Measure

The sound data was split into speech and noise parts by using the given transcription files and regardless of the type of noise. From the resulting data the MFCC feature vectors were calculated and used to train the separate GMMs for all noise and all speech. The framelength is set to 10 ms and the number of coefficients per frame is 12 plus energy. For the separation the difference in log likelihood $L(x)$ of the output probabilities of each frame was calculated and the histograms for speech-only and noise-only input were generated (Fig. 4, left). Now a threshold θ is chosen (see also section 4). Every frame with $L(x) \geq \theta$ is considered to be speech, otherwise noise.

$$L(x) = log \frac{P(x \mid speech)}{P(x \mid noise)} \tag{1}$$

The start-point for a speech section was set when the decision for the actual frame was speech and the previous frame was considered to be noise. For the endpoint it is vice versa.

3.4 Using Single Frames

The first approach was to calculate the GMMs with feature vectors that contained the information of one single frame. Due to the short window size and the missing context the data was cut into many small fragments. To improve this result the context on both sides has to be included. In order to do this but avoid calculation cost during testing the same GMMs and the same calculation per frame was used but $L(x)$ of n frames was summed up.

$$L(x_i) = \sum_{j=i-\frac{n}{2}}^{i+\frac{n}{2}} L(x_j) \tag{2}$$

This smoothened the result but the overall separation correctness did not improve as the resulting likelihoods accumulated in the same area. Thus reasonable separation was not possible.

3.5 Using n Frames

The second approach was to build up big feature vectors by combining the vectors of n frames including the energy. The important idea of this is not to use a block of n frames and then continue with the next n frames but to shift a window of n frames by one frame steps. Those *longvectors* are already used for the training of the GMMs and have to be built up for the evaluation of the test data, too.

The big disadvantage of this method is the high calculation cost. For each size of *longvector* which is $n * 13$ in this case, a new GMM has to be trained. But as this is only needed once for each set of training data this is acceptable. The creation of the *longvector* is fast and does not slow down the evaluation too much.

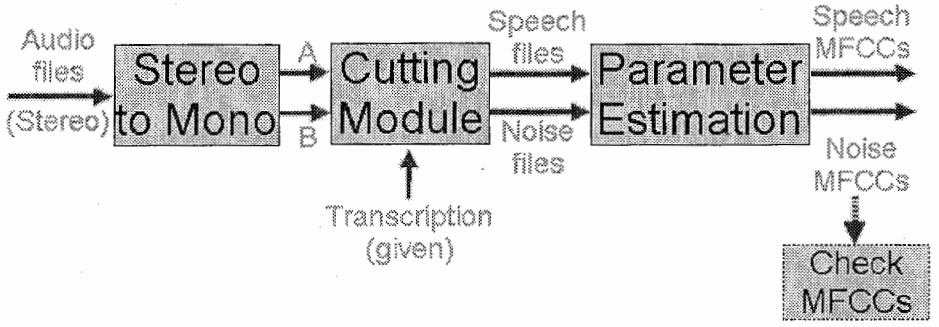

Figure 2: data preparation

3.6 Implementation

To be able to work with the data the stereo data has to be converted to mono. For the training of separate noise and speech GMMs the data also had to be cut. The information about the speech and noise parts was given in the transcriptions to each data file. Both tasks were realized with a PYTHON script. PYTHON allows the easy manipulation of strings and does not need compilation before running and was therefore chosen instead of a normal shell script or C/C++. From this mono speech and noise data *MEL frequency cepstrum coefficients (MFCC)* were calculated using the HTK toolkit. The module that offers this function is called HCopy. The input of this module is a list containing the names of all files from which the MFCCs should be calculated and a configuration file containing the settings for the feature vectors (see also section 3.2).

From this data the codebooks and then the GMMs were calculated. As already described in section 2 the software for the training of CBs and GMMs and a program for GMM testing exist. The proposed method is to decide according to the output probability of a feature vector if it represents speech or noise. For the decision the probability is compared with a threshold.

3.7 Realization of The Single Frames Method

For the method using single frames both the CBs and GMMs can be trained with the existing software, thus only the test software had to be modified. Until now the output probabilities for the whole test-data is calculated but for the future experiments the probability of each feature vector has to be calculated and written to a file instead of simple screen output. As all the functions needed are given only minor changes are necessary. Additionally I had to include a function for the decision as described in section 3.4.

Experiments showed that depending on the missing context of this method only single frames and very short segments of data were cut but not the whole speech parts like necessary for further recognition. An applied method to enhance the block size for the decision smoothened down this effect but did not improve the cutting in whole.

3.8 Realization of The Multi-Frame Method

Due to the previous results *longvectors* were introduced. They are built by combining n feature vectors to one vector i.e. $(n-1)/2$ left and the same number of right neighbor vectors are used to include more context dependency. A separate script generated these

longvectors from the features calculated with HTK. The selected number of frames was five or 25 frames.

The functions of the CB and GMM training software was designed to read all vectors into the memory before starting the calculation to have a faster access to this data. As this vectors were n times bigger than the previous used normal vectors the size of data increased with the same factor causing memory problems. Therefore a modification was made to read the vectors one after another from file. Reading from file is slower than accessing data from memory but in comparism to the calculation time this was acceptable.

In comparism to the single frame method the cutting improved but could not reach the level of the energy-based (EPD) method [JMR94]. Thus a combination of those two techniques was considered.

3.9 Energy Based Endpoint Detection

This method decides due to the energy amplitude of a signal. If this energy is above a set threshold the signal is considered to be speech. Another criterium is the speed of changes in the energy level. Slow or very fast changes are typical for noise, in between this for speed. The start and end-times of the cut noise are written into a file.

3.10 Combination Method

This approach uses the parts that are cut out with EPD and reviews the decision for these blocks. From the audio-data of a block the feature vectors are extracted and longvectors are built. For each longvector the output probability is calculated and the decision for speech or noise is done with the method explained in 3.5. The decision for the block is made by majority of all vectors in this block.

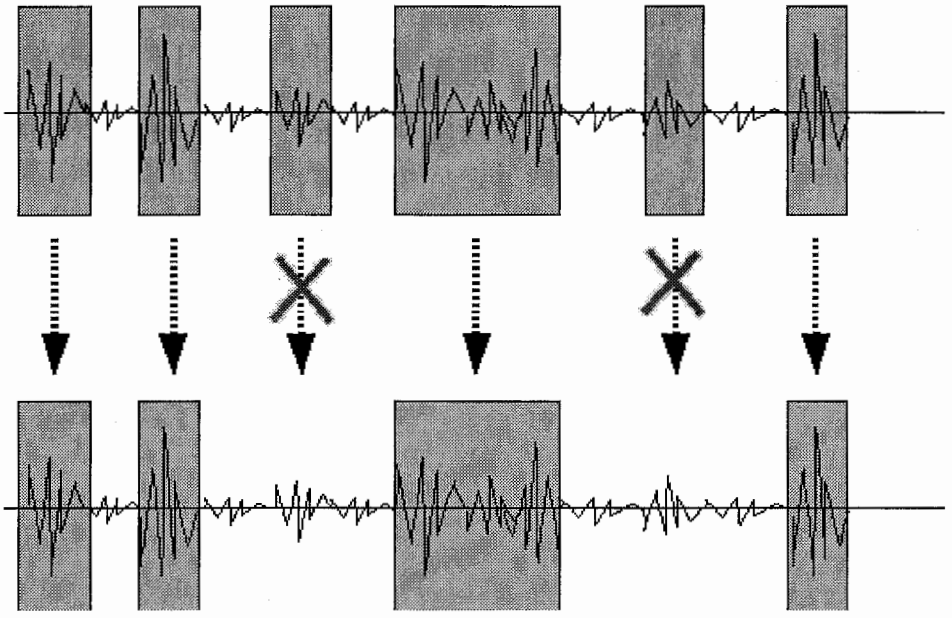

Figure 3: A short audio signal with the speech-parts cut on energy basis (first line) and after verification of these speech parts with the GMM-based method

4 Estimating The Threshold

As the value for the threshold is not known the first step is to estimate it. With the function to write the output probabilities into a file already implemented this data could be used to create a histogram i.e. a figure that shows which shows the distribution of the output probabilities. For the estimation the histograms for both the speech-only input and the noise-only input is needed. An example for 5-frames feature vectors for training and testing is shown in figure 4.

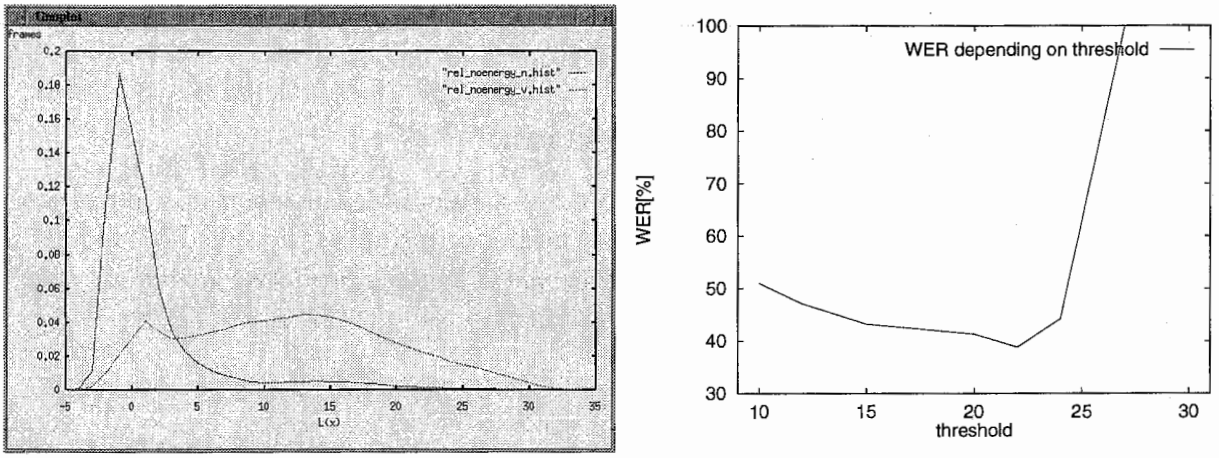

Figure 4: 5 frames training & test (left) and WER depending on the threshold (right)

There are some strategies for the final setting of the threshold for the decision of each frame. The first one is to minimize the noise by keeping as much speech as possible. The second one is to throw away most of the noise and accept loosing some speech parts, too. The results in figure 4 (right) show that latter approach gives the best results. If the type of noise is known this threshold can be chosen quite easily. For the SPINE task the evaluation noise types are unknown and one common threshold has to be used. Thus the difficulty is to find a threshold that fits all the noises. If the setting is too high most of the data will be considered to be speech and therefore the data will remain in large chunks including noise sections. A too low value results in cutting into tiny fragments. Especially in the latter case the recognition module will not be able to use an n-gram language model.

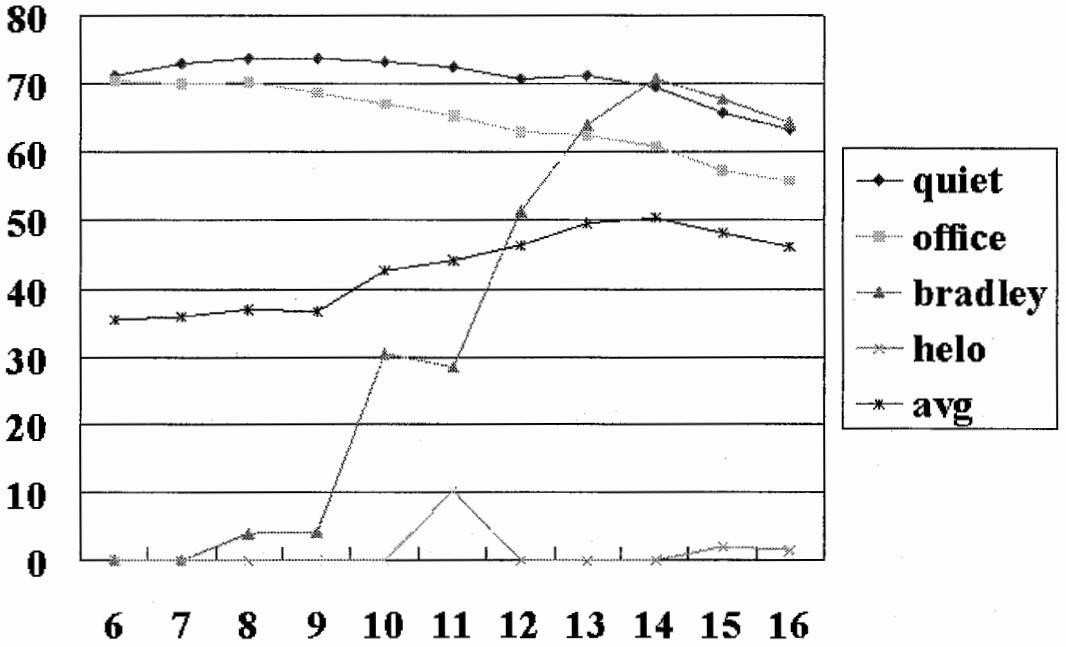

Figure 5: SPD performance (word accuracy) depending on threshold setting.

5 Method Comparison

As described in the previous chapters, the following three algorithms were applied to the SPINE 2 task:

EPD : an energy- and energy-derivation based approach

SPD : an algorithm using gaussian mixture models (GMMs) to make a framewise speech — non-speech decision

combination : EPD for basic speech area selection and SPD for blockwise confirmation

The approaches were evaluated on a subset of the SPINE 2 development data. Three conversation sides each for the four noise conditions clean, office, bradley (tank) and helo (helicopter). Among the multiple steps in the main evaluation, only the MFCC LTR feature was considered, the adaptation step skipped. The beam was set to 85,85.

Figure 5 shows the performance of the SPD approach for various threshold settings. The best threshold is different for each noise type, for clean conditions, a low threshold is important, for barley noise a high threshold performs better. An acceptable threshold for helo noise could not be found.

Figure 6 gives word accuracies for the various segmentation approaches and transcription-based segmentation depending on noise type. SPD performs best in silent condition. For moderate noise, all algorithms show similar performance. SPD completely fails for helo noise.

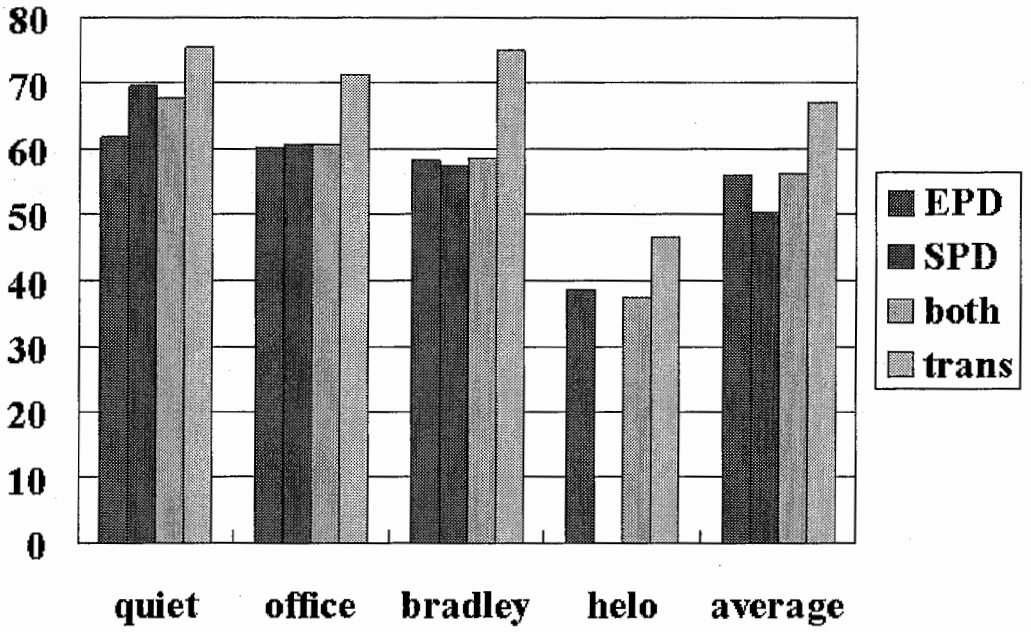

Figure 6: The word accuracy archived with various segmentation methods for (from left) clean condition, office, bradley and helo noise. The rightmost bars show the average performance.

Table 1 shows the word accuracy rates for each of the approaches. The combination method performs best among the automatic segmentations and reaches a fair 56.2% word accuracy considering the strong noise conditions. Still there is a gap of more than 10% to manual segmentation.

EPD	SPD	Combination	Transcription
56.0	50.3	56.2	67.1

Table 1: The word accuracy for the baseline separation according to transcription, the EPD and the combination method

A Software

A.1 Structure

The following diagrams show the modules of the separation system. In the compact version the modules to remove the HTK (or any other) header and the function to check the MFCCs are included into the module for the creation of the longvectors.

All datafiles have a postfix consisting of "." and three letters. When using two channels, additional ".A" or ".B" is added before (<name>.A.16k). When using noheader.py to remove headerbytes, the converted file.mfc is saved as file.nh.mfc.

Used postfixes are:	
.wav, .sph, .raw	stereo data
.16k	mono data
.mfc, .nh.mfc	MFCC files
.typ, .txt	transcription files

The software is designed for processing big-endian data on little-endian machines. If used in different conditions a careful examination of the byte-order is neccessary.

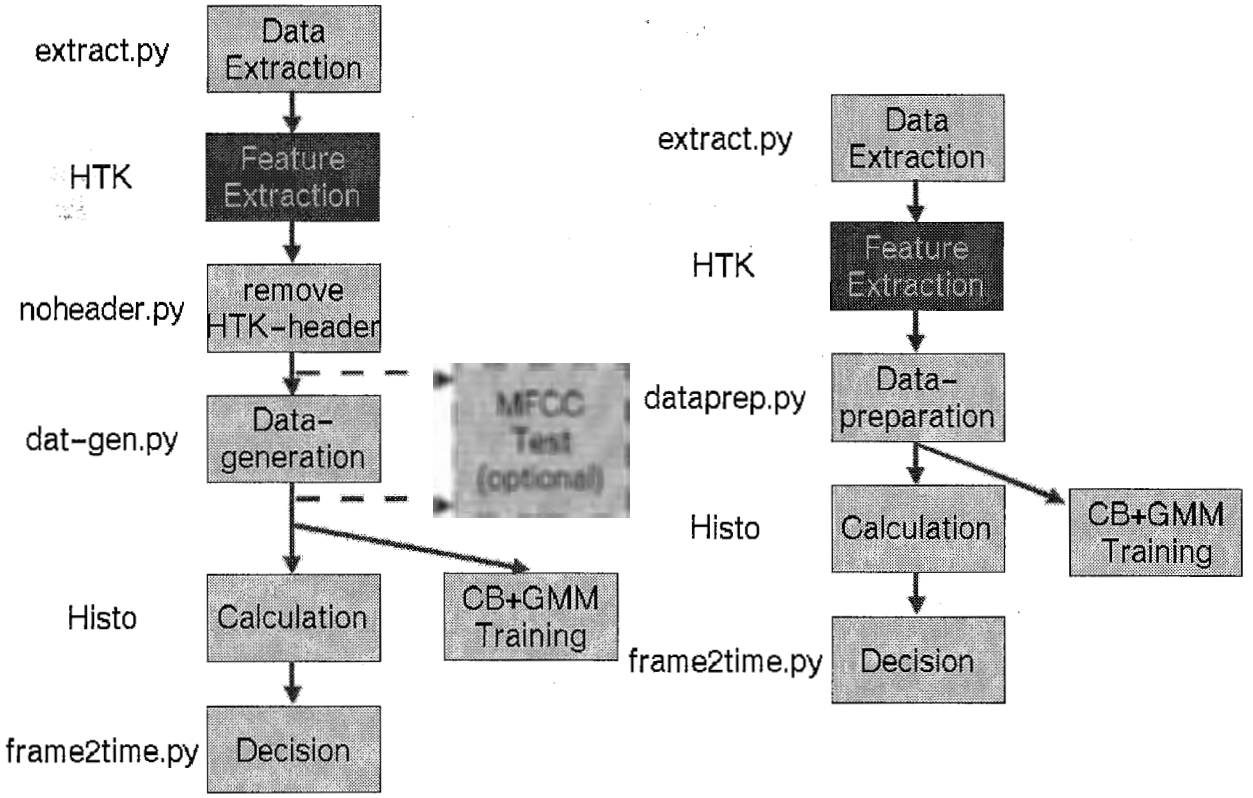

Figure 7: original system as used in SPINE2 evaluation (left) and compact version of system with noheader.py and mfcc-check.py included into datgen.py (right)

A.2 extract.py

Written by Rainer Gruhn.
It extracts the speech- or noise-parts of a file if the transcription is given in the format (where $\{A|B\}$ means channel A or B):
""""

$< starttime[sec] >< space >< endtime[sec] > \{A|B\}[< space >< text >]$
$< starttime[sec] >< space >< endtime[sec] > \{A|B\}[< space >< text >]$
[...]
""""

extraction.py <filelist><target-dir><type><stereo2mono> [<transcription-dir>]

filelist	format see above
target-dir	here, the data for both channels are saved; additional folders "speech" and "noise" for the separated data are created there, too!!!
type	{speech—noise}
stereo2mono	{TRUE—FALSE}; using TRUE splits up into 2 channels using FALSE, the channel has to be defined e.g.: <filename>.A.16k

e.g.: *python extraction.py /flist_s2dev /tmp/longwav/s2dev speech FALSE*

A.3 Feature Extraction With HTK

After extracting feature files can be created. For the task HTK [Y+99] was used. Start with:

HCopy -T 1 -C config -S filelist.scp

An example *config.htk* is included. A filelist (*.scp) containing source and target files is needed as input. The format is
""""

$< source.\{wav|raw|16k|...\} >< SPACE!!! >< target.mfc >$
$< source.\{wav|raw|16k|...\} >< SPACE!!! >< target.mfc >$
[...]
""""

A.4 noheader.py

HTK includes a header of 12 byte which has to be removed before using the MFCC-files. The target mfccs are labeld with .nh.mfc and written in the same directory. The names of the files to modify have to be written in a list containing the whole path! The name of the list can be modified (line: IDlist="/home/someone/put/it/in/here/list")

A.5 mfcc-check.py

Sometimes the MFCC-files seemed to be corrupted. Some values of the created files where huge. This skript checks the data and writes out the values which are extraordinary high.

python mfcc-check.py < listname >

< listname >	filelist
returns:	*< listname > .checked* for the good and *< listname > .error* for the bad files.

<! > This is only a rough check but it worked fine for the given data!!!

A.6 datgen.py

If not only single MFCCs are needed but longvectors consisting of n vectors this can be done them with this skript:

python datgen.py <L><R><Dim><List><ign. left><ign. right> [<out-file>]

L	left context
R	right context
Dim	dimension of the (single) feature vector
ign. left	skip first x entries of a feature vector (e. g. energy)
ign. right	skip last x entries of a feature vector
out-file	name for output file (default: multiframes) the ending is .mfc

Example: for vectors of 5 frames (2 left and 2 right neighbours context) and MFCCs with a dimension of 13 and not ignoring energy or other features this is:

python datgen.py 2 2 65 flist_mfcc 0 0

As no output-file is defined, this is set to *multiframes*. If the number of inputfiles is higher, more than one file is created. The names of the created files are written in <out-file>.list.

A.7 frame2time.py

The results of "Histo" is a list of values separated by LF. Depending on a threshold bigger values are considered to be speech, smaller noise. As the decision is too rough it is smoothed by a majority-decision over 25 frames (this thresh can also be shifted but the switch is in the skript:
$if dec > count* < value >$; value=0.5 was too hard!)

pythonframe2time.py < srcFile >< targetFile >< Thresh >< framelength >

srcFile	file containing the differences in log-likelihood separated by "LF" e. g. rel.data from Histo
targetFile	file in the format: <start><end><typ> typ: "n" noise, "s" speech
Thresh	threshold for decision between speech and non-speech
framelength	framelength in ms

A.8 dataprep.py

This skript combines noheader.py, mfcc-check.py and datgen.py into one. The options for the start are the same than for datgen.py. The number of headerbytes is set to 12 as used in HTK (HCopy) but it can be changed (in the header of the skript: *htk_header=12 # HTK-header in MFCCs*) These bytes are removed during reading. The check of the MFCC vectors is done before creating the longvectors. If problems appear with the longvectors, they can be checked with mfcc-check.py as usual.

A.9 histo_gen.py

The outputfile from Histo can be used to generate a histogram. histo_gen.py reads a file (normally with the ending <file>.data) and saves the data for the histogram in <file>.hist. The file can be plotted e. g. with gnuplot. This is very helpful for speech-only and noise-only input. When printed in one histogram, the threshold can be chosen more easily.
The structure of the outputfile is:
$< x - value >< amount >$
$< x - value >< amount >$
[...]

python histo_gen.py <histo.data> [<normalization>]

histo.data	filename of inputfile
	single values separated by LF
normalize	value to normalize the histogram
	e. g. number of frames

A.10 CB- and GMM-training

The original software was written by Konstantin Markov. I modified it so it can read longvectors. The problem is as the memory is too small, it can not read them into memory at once but has to read them from file. The calls for the programs are the same like before. Please take care to enter the right dimension when using longvectors!

(A.10.1) TrainCB

This Program creates a codebook from vector data.

TrainCB.spu <vec_flist><cb_file> [Options]

-l	<codebook level> (Default - 64)
-d	$< n >$ ... input vectors dimention. (Default - 11)
-w	... to use WLR distance (Default - CEPSTRAL)
-r	$< n >$ ... to use RGC with width of $< n >$
-n	<number of iterations> (Default - 10)
-s	... to save intermediate codebooks
-p	... to print Statistical data

(A.10.2) TrainGMM

This program trains a GMM from vector data in combination with a codebook trained with TrainCB.

TrainGMM.hp <vec_flist><cb_file><gmm_file> [Options]

-n	<number of iterations> (Default - 10)
-s	... to save intermediate gmms
-p	... to print Statistical data
-d	... to use diagonal matrices (Default - full)
-m	<data multipl.coefficient> (Default - 100)
-i	... to save initial model

(A.10.3) Histo

The log likelihood for the input data is calculated with each GMM and for each input vector the difference in log-likelihood is calculated. This is written in the file rel.data. The so created data can be used to generate a histogram or for the speech—non-speech separation module.
Usage: *Histo <gmm_list><test_list><hist_file>*
Returns: rel.data

the resulting file is rel.data which contains the difference in log likelihood calculated from the noise and speech GMM.

B Evaluation Results

File	Noise	Word accuracy [%]													
		Combination			GMM							EPD			Trans
		11beam85	11beam88	Thr9	Thr6	Thr7	Thr8	Thr9	Thr10	Thr11	Thr12	std	silent	loud	
g07a01	office	49.3	47.4	49.3	61.2	61.2	61.2	60.5	57.9	55.3	50.0	48.0	49.3	31.6	63.2
g07a01	quiet	70.6	68.7	70.6	71.2	71.2	73.0	73.0	73.0	73.0	71.8	70.6	65.6	67.5	76.1
g07a02	office	59.8	43.2	59.8	65.7	65.1	65.1	63.3	63.3	62.1	58.6	59.2	53.3	41.4	66.9
g07a02	quiet	70.9	69.5	70.9	76.8	77.5	78.1	78.8	78.8	76.8	75.5	71.5	61.6	65.6	80.1
g07a03	office	73.6	72.8	73.6	84.8	84.0	84.8	82.4	80.0	78.4	80.8	73.6	67.2	77.6	84.0
g07a03	quiet	62.0	52.9	62.0	66.1	70.2	70.2	69.4	67.8	67.8	65.3	59.5	48.8	45.5	70.2
g07d01	bradley	54.7	44.4	54.7	---	---	3.4	3.4	20.5	28.2	48.7	54.7	32.5	57.3	77.8
g07d01	helo	25.8	18.3	25.8	---	---	---	---	---	23.8	---	29.0	1.1	28.0	36.6
g07d02	bradley	54.5	49.0	54.5	---	---	---	---	25.2	52.6	42.8	53.1	36.4	58.0	71.3
g07d02	helo	48.2	42.0	48.2	---	---	---	---	---	6.8	---	48.2	25.0	42.9	55.4
g07d03	bradley	66.2	58.6	66.2	---	---	9.0	9.8	45.9	4.2	62.4	66.9	36.1	71.4	78.9
g07d03	helo	38.6	43.6	38.6	---	---	---	---	---	0.0	---	38.6	8.9	33.7	---

SPINE2
Best result
---: sclite failed

Figure 8: Evaluation results for the development data of the SPINE2 database

References

[BMGN01] Norbert Binder, Konstantin Markov, Rainer Gruhn, and Satoshi Nakamura. Speech—Non-Speech Separation with GMMs. *Proc. Autumn Meeting Acoust. Soc. Jap.*, pages 141–142, 2001.

[JMR94] J-C Junqua, B Mak, and B Reaves. A robust algorithm for word boundary detection in the presence of noise. *Proc. IEEE*, 2(3):406–412, 1994.

[MMG+01] Konstantin Markov, Tomoko Matsui, Rainer Gruhn, Jingdong Chen, Kaiseng Yao, and Satoshi Nakamura. 2000 DARPA SPINE1 evaluation. Technical Report TR-S-027, ATR-SLT, June 2001.

[SSRS01] R Singh, M Seltzer, B Raj, and R Stern. Speech in noisy environments: Robust automatic segmentation, feature extraction and hypothesis combination. In *Proc. of ICASSP2001*, 2001.

[Y+99] S Young et al. *The HTK Book.* Entropic Ltd, 1999.