Hidden Markov Tool Kit

An Introduction

University Magdeburg - Kognitive Systeme

Contents

Basics:

- What is HTK, what is it not?
- Structure of HTK
- The most important HTK-modules

Example:

- Isolated Word Recognition of digits

University Magdeburg - Kognitive Systeme

What is HTK ?

- Toolkit for building Hidden Markov Models
- Used for researching projects like
  - Speech recognition, handwriting recognition, face recognition
- Developed at the Cambridge University Engineering Department (CUED)
  - Free available: http://htk.eng.cam.ac.uk/
  - Sourcecode in C
  - Developed on Unix/Linux mashines
  - In the meantime also on Windows systems useable

What is HTK not?

- HTK is not a desktop dictation system
- In general no graphical in- and outputs are used with HTK
- It contains no own database (like speech, text or lexica)

➢ Usage:

researching and developing of new algorithms, trainingstrategies, etc. !

Structure of HTK

- Organisation:
  - The toolkit is divided in several modules, e.g.
    - Feature extraction
    - Parameter estimation (training)
    - Recognition
- Call of program module:
  - Command line (single modules)
  - Shell-script (several modules successively)
  - mhtk (comfortable configuration environment, but absolutely beta!)

The main HTK-modules

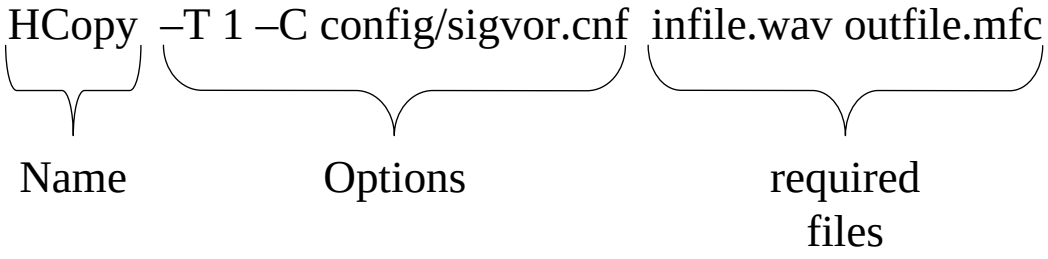
- HCopy (feature extraction)
  - ➔ signal processing
    (Windowing, FFT, LPC, Cepstrum, Deltas)
- HERest (parameter estimation)
  - ➔ Estimation of the HMM-parameter using Baum-Welch
- HVite (recognition engine)
  - ➔ Estimation of the most likelihood hypothesies from given feature vectors and a language model

The main HTK-modules

- HHEd (edits HMMs)
  - ➔Production of mixture densities, HMM cloning
- HLEd (edits label files)
  - ➔Transcribtion of given text
- HDMan (generates new lexica)
  - ➔Triphon-lexicon from monophon-lexicon

General call of modules

- Example: signal processing using HCopy:



- Disadvantages:
  - Lots of different options/configurations
  - Difficult to reproduce, nontransparent
- Solution:
  - Module-call by Perl-scripts with a clear configuration enviroment (mhtk)

General call of modules

File **sigvor.cnf** contains all required configurations for signal processing:

```
SOURCEKIND = WAV
SOURCERATE = 625
WINDOWSIZE = 250000
TARGETRATE = 100000
NUMCEPS    = 12
...        = ...
```

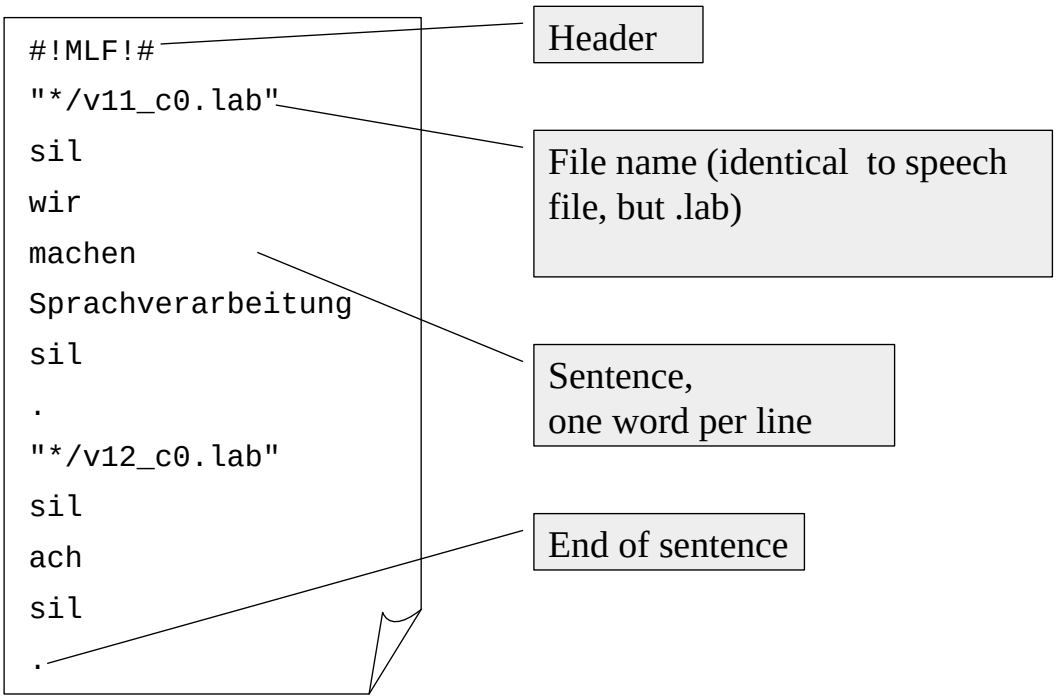
Taking down al lot of speech files, we can put them in one script:

```
/data/sa1.wav   /data/sa1.mfc
/data/sa2.wav   /data/sa2.mfc
/data/sa3.wav   /data/sa3.mfc
/data/sa4.wav   /data/sa4.mfc
...             ...
```


Requirements

- Speech files
  - advantageous : one sentence per file
- Respective reference sentences
  - Should be written in a so called Master Label File (MLF)
- Lexicon
- Prototype of Hidden Markov Models
  - How many states, which dimension, which transitions should be used

Master Label File (MLF)



Lexicon

- phonetical transcribtion of all used words (e.g. IPA, SAMPA):

Monophone:

eins	aI n s
zwei	ts v aI
...	

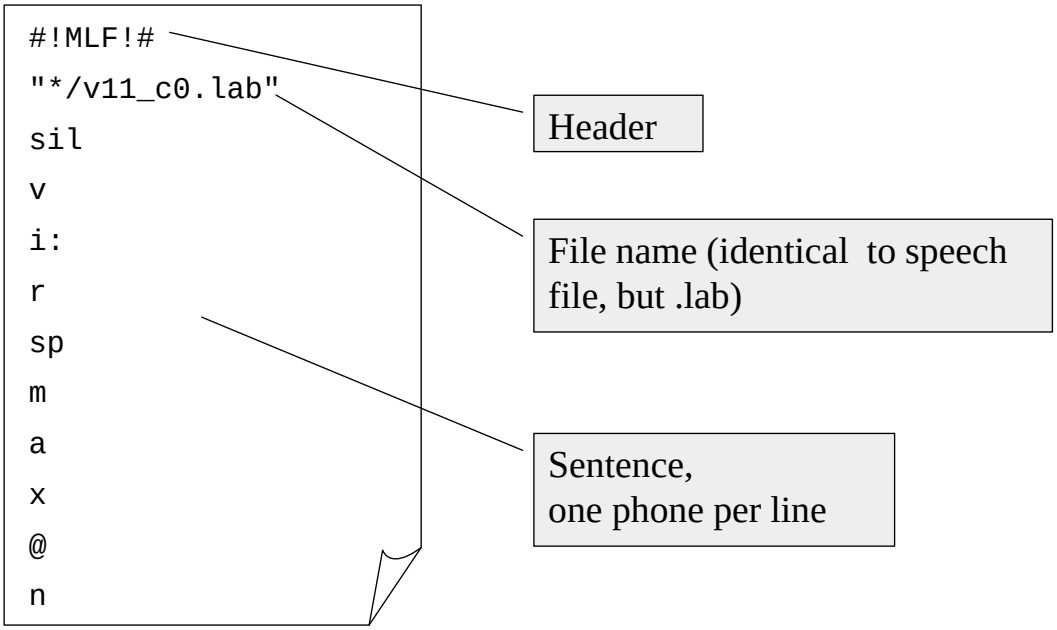
Triphone:

eins	aI+n aI-n+s n-s
zwei	ts+v ts-v+aI v-aI
...	

- The training lexicon must contain all words of the training sentences
- For recognition only the words in the recognition-lexicon could be recognized

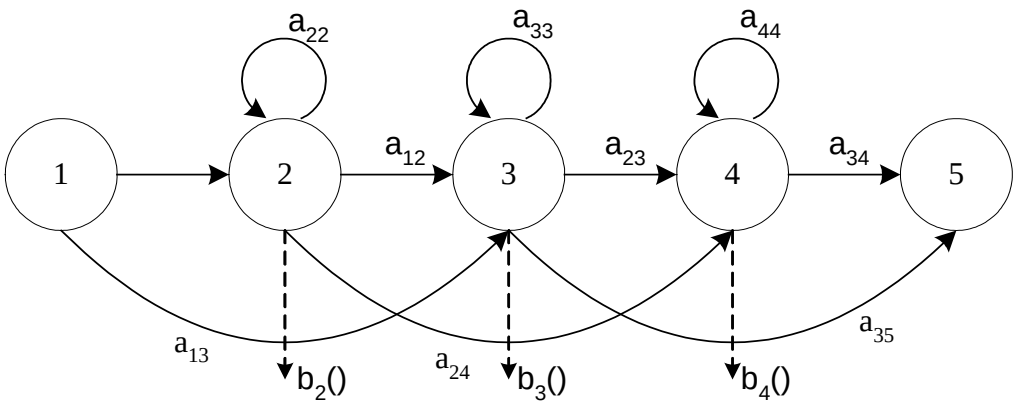
Master Label File (MLF)

- HTK-Call for transcribtion:
  HLEd –I input.mlf –i output.mlf –l '*' –d phon.dic editfile



Hidden Markov Model

- We are looking for that HMM, which produced the given feature sequence



Hidden Markov Model

- The emisson probability:

$$b_j(o_t) = \sum_{s=1}^{M_j} N(o_t; \mu_{jm}, \Sigma_{jm})$$

- Distribution:

$$N(o_t; \mu_{jm}, \Sigma_{jm}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)^{tr} \Sigma^{-1} (o-\mu)}$$

- Summarizing the transition probabilities in a N x N-matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix}$$

HMM-Prototype

```
~h "proto"
<BeginHMM>
<VecSize> 39 <MFCC_E_D_A>
<NumStates> 5
  <State> 2
  <Mean> 39
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  <Variance> 39
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 3
  …
  <State> 4
  …
  <TransP> 5
    0.000e+0   1.000e+0   0.000e+0   0.000e+0   0.000e+0
    0.000e+0   5.000e-1   4.000e-1   1.000e-1   0.000e+0
    0.000e+0   0.000e+0   5.000e-1   4.000e-1   1.000e-1
    0.000e+0   0.000e+0   0.000e+0   5.000e-1   5.000e-1
    0.000e+0   0.000e+0   0.000e+0   0.000e+0   0.000e+0
<EndHMM>
```

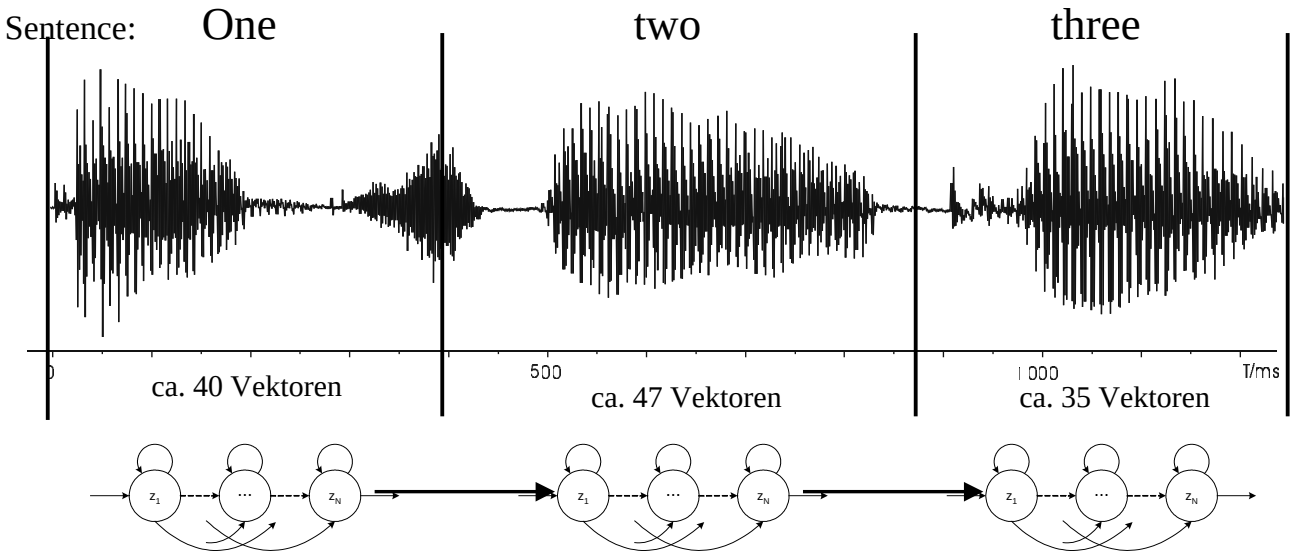
Estimation of HMM-parameter

- Time aligned training data by HInit
- Training data without time alignment by HERest
  - Concatenation of all used HMMs per sentence
  - One call of HERest will check the whole data
    - The parameter of class-means, variances and transitions have to be computed
    - Update of the HMMs after every pass

Estimation of HMM-parameter

- Concrete:
  - Gaussians from feature vectors
  - Transition probabilities from gaussians
  - Calculating forward-backward matrices
  - Better estimation of the means and the variances from the forward-backward matrices
  - Pass several iterations (until convergence)

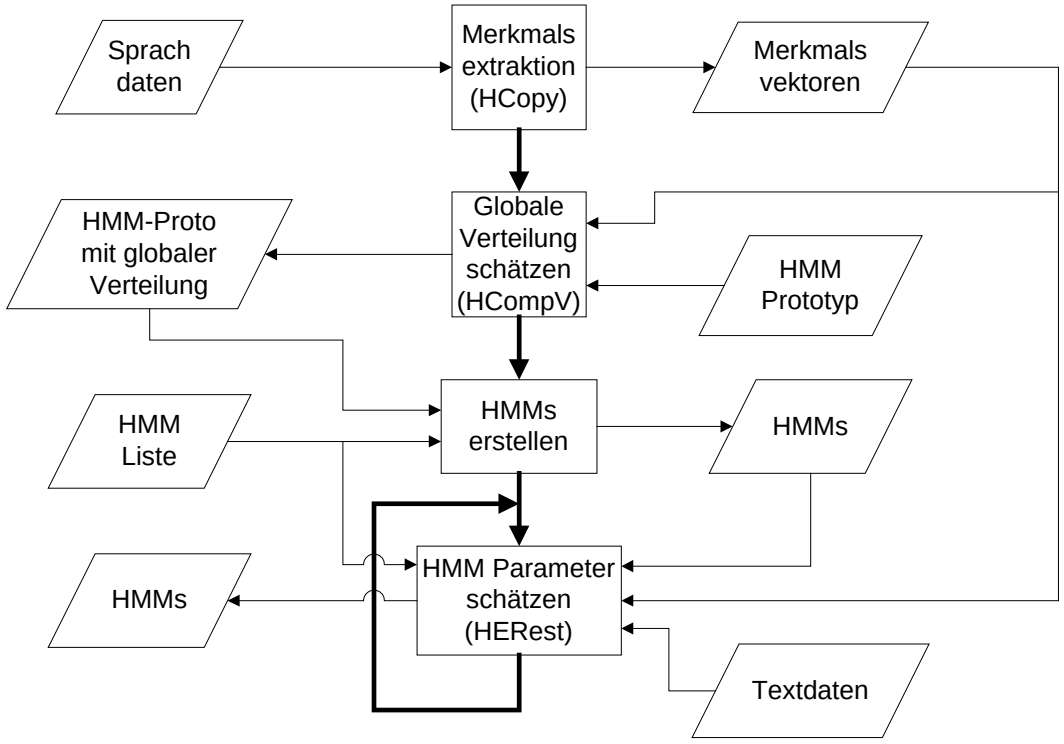
Estimation of HMM-parameter



Questions:

- How many states we need?
- How many iterations we need?

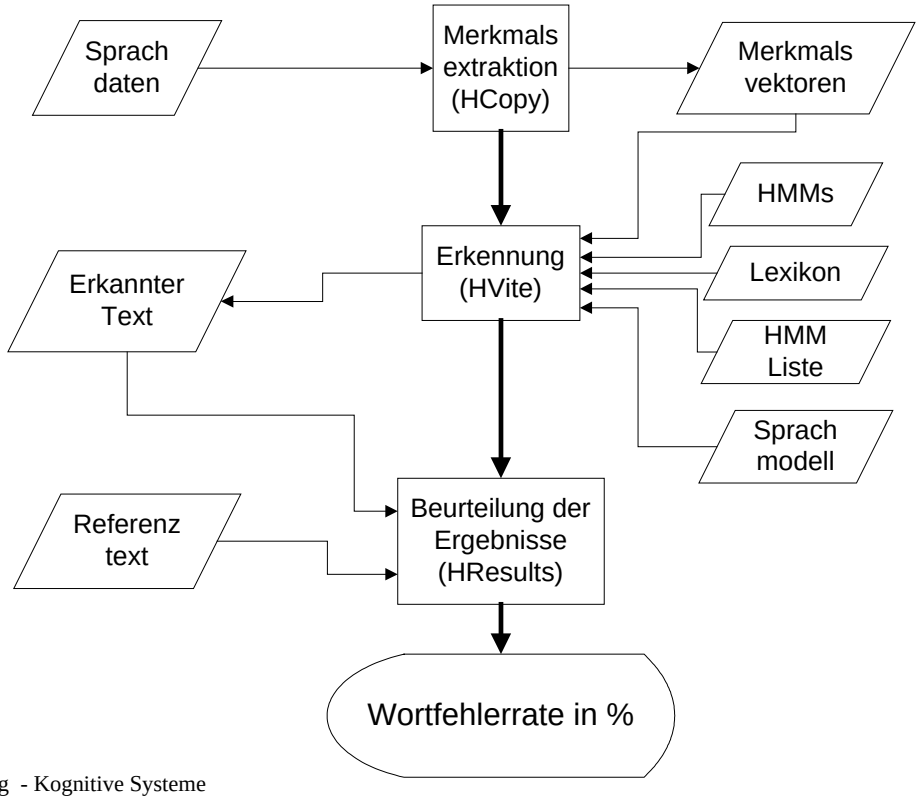
Operating sequence (Training)



Speech recognition using HTK

- For recognition (HVite) we need:
  - Speech data (Testdata)
  - trained Hidden Markov Models
  - Lexicon
  - Language Model
- Evaluation of the results (HResults):
  - Resulting sentences from HVite
  - Reference text of the speech data

Operating sequence (Test)



Recognition (HVite)

- Call:

  HVite –H modelle/hmm8/models –S lists/test.feat –w net/wordnet –i test/results dict/ziffern.dic lists/wordlist

  recognized text (MLF)

- Presentation of the most likely hypothesies in a so called Lattice
- Calculating the most likely path thru the lattice using the Viterbi-algorithm

Viterbi-recognition

```
VERSION=1.0
UTTERANCE=0001
lmname=net/net
lmscale=0.00 wdpenalty=0.00
vocab=dict/word_sp.dic
N=24   L=40
1   I=0    t=0.00  W=!NULL
2   I=1    t=0.46  W=sil
    I=2    t=0.78  W=fuenf
    I=3    t=0.79  W=sp
    I=4    t=0.81  W=fuenf
    I=5    t=0.90  W=fuenf
    I=6    t=0.91  W=acht
3   I=7    t=0.96  W=fuenf
    I=8    t=0.96  W=sp
    I=9    t=1.03  W=fuenf
    I=10   t=1.03  W=sp
    I=11   t=1.41  W=drei
    I=12   t=1.42  W=drei
    I=13   t=1.42  W=sp
    I=14   t=1.74  W=eins
    I=15   t=1.75  W=fuenf
    I=16   t=1.75  W=eins
    I=17   t=1.76  W=sp
4   I=18   t=1.76  W=eins
    I=19   t=1.77  W=sp
    I=20   t=1.93  W=acht
    I=21   t=1.94  W=sp
6   I=22   t=1.94  W=!NULL
5   I=23   t=1.94  W=sil
```

```
J=0     S=23   E=22   a=0.00       l=0.000
J=1     S=18   E=23   a=-1176.62   l=0.000
J=2     S=8    E=18   a=-5316.18   l=0.000
J=3     S=5    E=8    a=-499.34    l=0.000
J=4     S=1    E=5    a=-3070.80   l=0.000
J=5     S=0    E=1    a=-3163.98   l=0.000
J=6     S=6    E=8    a=-417.77    l=0.000
J=7     S=2    E=6    a=-970.12    l=0.000
J=8     S=1    E=2    a=-2259.74   l=0.000
J=9     S=3    E=6    a=-904.06    l=0.000
J=10    S=2    E=3    a=-70.10     l=0.000
J=11    S=7    E=18   a=-5316.18   l=0.000
J=12    S=1    E=7    a=-3570.31   l=0.000
J=13    S=4    E=7    a=-1198.35   l=0.000
J=14    S=1    E=4    a=-2446.81   l=0.000
J=15    S=17   E=23   a=-1176.62   l=0.000
J=16    S=16   E=17   a=-80.13     l=0.000
J=17    S=8    E=16   a=-5238.19   l=0.000
J=18    S=7    E=16   a=-5238.19   l=0.000
J=19    S=15   E=17   a=-80.13     l=0.000
J=20    S=13   E=15   a=-2140.15   l=0.000
J=21    S=11   E=13   a=-72.70     l=0.000
J=22    S=9    E=11   a=-2511.00   l=0.000
J=23    S=1    E=9    a=-4133.05   l=0.000
J=24    S=4    E=9    a=-1759.71   l=0.000
J=25    S=12   E=15   a=-2140.15   l=0.000
J=26    S=9    E=12   a=-2583.86   l=0.000
J=27    S=18   E=19   a=-79.10     l=0.000
J=28    S=15   E=19   a=-159.40    l=0.000
J=29    S=10   E=11   a=-2511.00   l=0.000
J=30    S=10   E=12   a=-2583.86   l=0.000
J=31    S=21   E=22   a=0.00       l=0.000
J=32    S=5    E=10   a=-1062.08   l=0.000
J=33    S=6    E=10   a=-980.50    l=0.000
J=34    S=8    E=14   a=-5168.86   l=0.000
J=35    S=7    E=14   a=-5168.86   l=0.000
J=36    S=18   E=21   a=-1229.28   l=0.000
J=37    S=20   E=21   a=-57.70     l=0.000
J=38    S=19   E=20   a=-1134.11   l=0.000
J=39    S=14   E=20   a=-1358.37   l=0.000
```

Evaluation (HResults)

- Output of the word error rate:
  - I.e. acustical input is „recognized“ by HTK (HVite) and will be compared with the reference text

```
# ---------------------- Overall Results -------------------------
# SENT: %Correct= 0.00 [H= 0, S= 3, N= 3]
# WORD: %Corr= 63.91, Acc= 59.40 [H= 85, D= 35, S= 13, I= 6, N= 133]
# ============================================================
```

$$\text{Acc} = \frac{N - D - S - I}{N} \cdot 100\%$$

$$\text{Corr} = \frac{N - D - S}{N} \cdot 100\%$$

Total Label Numbers

Insertion Errors

Substitution Errors

Deletion Errors

Hit

Digits Recognition

We can‘t believe, so we have to investigate !!!

Requirements

- We need:
  - Speech data
  - Reference texts
  - Lexicon
  - HMM Prototype
  - Language Model [only for recognition]
  - Diverse Lists

Lists with Content

- Wordlist:

  Contains all used words

  (one word per line, in this examples the digits 0 ... 9)
- Lexicon (very simple here):

  zero zero sp
  one one sp
  ... ...
- HMM list:

  Contains the names of all used HMMs

  (here: HMM lists = wordlist)

File-Lists

- Speech data :

  ls data/*/train* > lists/wav.train

  /data/katz/train1.wav
  /data/katz/train2.wav
  ...

- Speech data -> feature vectors:

  gawk –F. ‘{print$0“ “$1“.mfc“}‘ lists/wav.train > lists/feat.train

  /data/katz/train1.wav /data/katz/train1.mfc
  /data/katz/train2.wav /data/katz/train2.mfc
  ... ...

File-Lists

- Feature vectors:

  gawk –F. ‘{print$1“.mfc“}‘ lists/wav.train > lists/feat.train

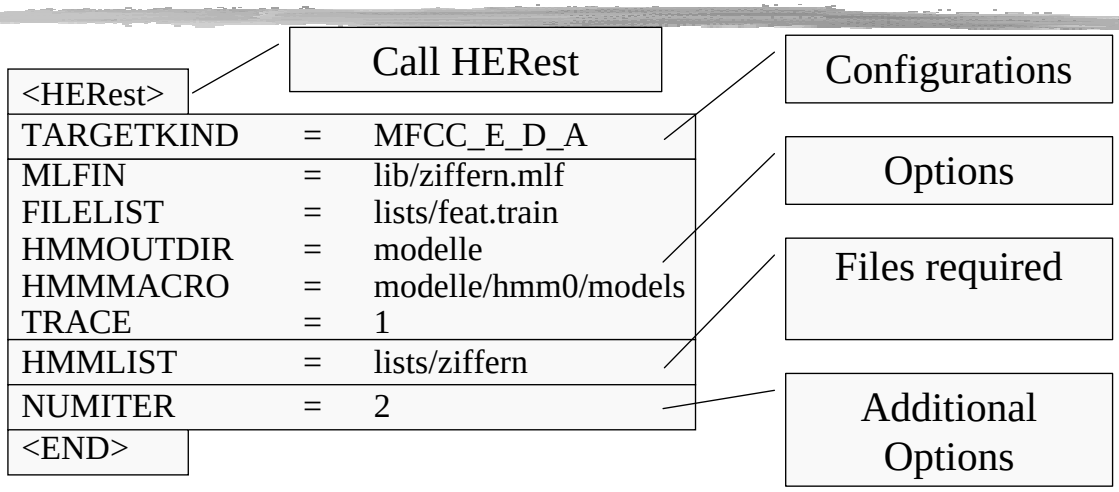
  /data/katz/train1.mfc
  /data/katz/train2.mfc
  ...

- Texts:

  gawk –F. ‘{print$1“.lab“}‘ lists/wav.train > lists/feat.train

  /data/katz/train1.lab
  /data/katz/train2.lab
  ...

The mhtk-Configuration environment



Outcome of this is the HTK-Command:

HERest –T 1 –C tmp/HERest.cnf –i lib/monophon.mlf –S lists/feat.train –H modelle/hmm0/models –M mod/hmm1/models lists/monophones

HERest –T 1 –C tmp/HERest.cnf –i lib/monophon.mlf –S lists/feat.train –H modelle/hmm1/models –M mod/hmm2/models lists/monophones

Recognition Live

- direct HTK-Command using a configuration file
  - HVite -A -C lib/hmm.live -p 0.0 -w lib/uni.net -H modelle/hmm1.8/models -T 1 -s 0 lib/word.dic
  - configs/hmm.live:

```
TARGETKIND    = MFCC_E_D_A
HNET:TRACE    = 1
SOURCERATE = 625
SOURCEKIND = HAUDIO
SOURCEFORMAT = HTK
TARGETRATE = 100000
USESILDET = T
MEASURESIL = F
OUTSILWARN = T
ENORMALISE = F
```

Recognition Live

- Configuration file for mhtk.prl:

```
<HVite>
    TARGETKIND     = MFCC_E_D_A
    SOURCERATE = 625
    SOURCEKIND = HAUDIO
    SOURCEFORMAT = HTK
    TARGETRATE = 100000
    USESILDET = T
    MEASURESIL = F
    OUTSILWARN = T
    ENORMALISE = F
    NETWORK = lib/uni.net
    HMMMACRO = modelle/hmm1.8/models
    INSPROB = 0.0
    LMSCALE = 0
    MLFOUT = test/word.test.mlf
    LEXICONIN = lib/word.dic
<END>
```