CONCEPTS OF THE DISSERTATION

Martin Karafiát

May 2003

Contents

1 Introduction

Since my diploma thesis [8] I have been oriented to digital speech processing, especially to speech recognition. My Phd studies are still concentrated in this area and I have recently started to focalize on automatic speech recognition domain (shortly ASR). This paper presents what I have done and what I would like to do until the end of my studies.

This paper is divided into 6 sections. Section 2 has two parts: the first is an overview of basic methods for processing and modeling acoustic features of speech. The second one is an introduction in large vocabulary speech recognition. Main types of speech decoders and problematic of language modeling is covered in this part. Section 3 introduces the problematic of recognition of meeting data in the frqmework of European project M4 Multi-Modal Meeting Manager.

Next, section 4 covers already finished experiments and a descriprion of the current ones. Last two sections: *Conclusions* 6 and *Plans* 5 summarize briefly already performed work and the plans till the end of my PhD studies.

2 Introduction in speech recognition

Speech recognition system can be represented by a black box converting acoustic signal to sequence of symbols, mostly words. When we open it, we will find out that almost all speech recognition systems consist of three main parts:

1. **Signal processing** - Extraction of features from speech waves by removing redundant information. Usually based on spectral analysis followed by some decolleration transform.

2. **Acoustic modeling** - Transformation a flow of features to sequence of symbols and probabilities.

3. **Decoding** - Generation of output sequence of words with using output information from *Acoustic modeling* and language knowledge.
   Note. This part is often nearly tight with *Acoustic modeling*.

Simple example of Speech recognition process is displayed in figure 1.

2.1 HIDDEN MARKOV MODELS

2.1.1 Introducion to HMMs

This chapter outlines what a hidden Markov Models (HMM) are and how it work. A hidden Markov Model is a stochastic model composed of series of connected states receiving discrete time signal and generate a probability of how model was covered with input sequence.
In each time step the model changes state according to *Transition probabilities* and resulting state generates a single *observation* according to *output probability distribution* of that state.
Each model has two unique states (first and last state) which do not have attached function of probability distribution and they are used for connecting of models.

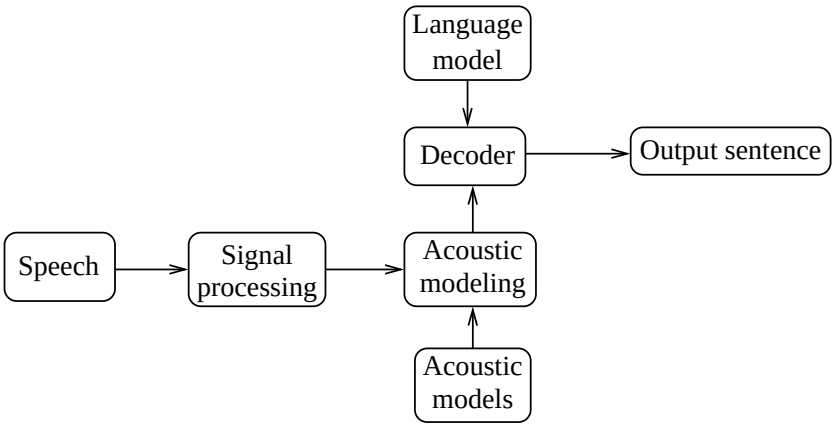

Figure 1: Example of recognition process.

The *output probability distribution function* $b_j(\boldsymbol{o}_t)$ describes the distribution of observation produced by state j. It produces the probability (discrete output) or likelihood (continuous output) of state j generating the observation $\boldsymbol{o}_t$.
Discrete probability distribution:

$$b_j(o_t) = \prod_{b=1}^{B} p_{jb}[v_b(\boldsymbol{o}_t)] \tag{1}$$

where $v_b(\boldsymbol{o}_t)$ is the output of vector quantizer using codebook b of observation $\boldsymbol{o}_t$ and $p_{jb}[v]$ is probability of state j generating symbol v from stream associated with b.
Continuous probability distribution is common Gaussian probability distribution:

$$b_j(\boldsymbol{o}_t) = N(\boldsymbol{o}_t, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\boldsymbol{o}_t - \boldsymbol{\mu}_j)' \boldsymbol{\Sigma}_j^{-1} (\boldsymbol{o}_t - \boldsymbol{\mu}_j)} \tag{2}$$

Where $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ is mean and covariance of Gaussian distribution $b_j(\boldsymbol{o}_t)$.

Transition probabilities a_{ij} describes of probability transition from state i to state j.
Example of hidden Markov Model is on figure 2. Where a_{ij} are transition probabilities

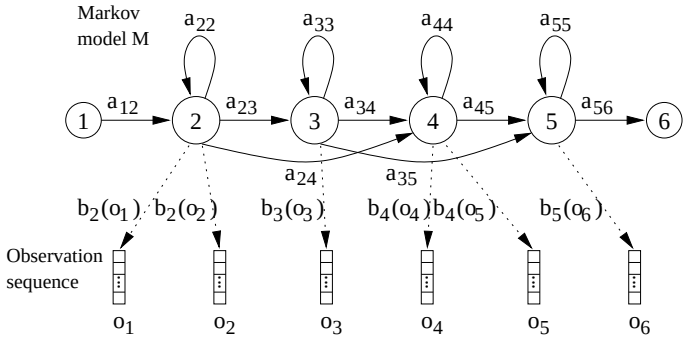

Figure 2: Hidden Markov Model.

between states, $\boldsymbol{o}_{1-6}$ are feature vectors and $b_j(\boldsymbol{o}_t)$ are outputs probability distributions.

So, model can be used to calculate the likelihood of a signal **O** consisting of T frames, $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$, being produced by a particular sequence of states, $X = x(1), \ldots, x(T)$.

For example the particular sequence in the figure 2. is
$X = 12234456$ so,

$$P(O|X) = a_{12}b_2(\boldsymbol{o}_1)a_{22}.b_2(\boldsymbol{o}_2)a_{23}.b_3(\boldsymbol{o}_3)a_{34}.b_4(\boldsymbol{o}_4)a_{44}.b_4(\boldsymbol{o}_5)a_{45}.b_5(\boldsymbol{o}_6)a_{56} \tag{3}$$

Or more generally,

$$P(O|X) = a_{1x(1)}b_{x(1)}(\boldsymbol{o}_1)\left(\prod_{t=2}^{T} a_{x(t-1)x(t)}b_{x(t)}(\boldsymbol{o}_t)\right)a_{x(T)N}. \tag{4}$$

In practical using of hidden Markov Models this state sequence is not known thus from here is derived name *hidden*. Alternatively, there are used likelihoods calculated for all states sequences (Baum-Welsh probability)

$$P(O, X) = \sum_X a_{1x(1)}b_{x(1)}(\boldsymbol{o}_1)\left(\prod_{t=2}^{T} a_{x(t-1)x(t)}b_{x(t)}(\boldsymbol{o}_t)\right)a_{x(T)N} \tag{5}$$

or state sequence which generates the highest likelihood (Viterbi probability).

$$\tilde{P}(O, X) = \max_X \left\{ a_{1x(1)}b_{x(1)}(\boldsymbol{o}_1)\left(\prod_{t=2}^{T} a_{x(t-1)x(t)}b_{x(t)}(\boldsymbol{o}_t)\right)a_{x(T)N} \right\}. \tag{6}$$

2.1.2 Recognition by HMM

In case of unknown sequence of observation O and set of models W the recognition is represented as the best match $\tilde{W}$ from the set of all possibly elections:

$$\tilde{W} = \arg\max_W \{P(W|O)\} \tag{7}$$

But from equation 4. there is possibly to calculate by HMMs only the best match sequence of observation to model. This problem could be solved by application by Baysian rule:

$$P(W|O) = \frac{P(W|O)P(W)}{P(O)} \tag{8}$$

Probability of observation $P(O)$ is constant thus from both equations above (7,8) is possibly to write.

$$\tilde{W} = \arg\max_W \{P(O|W)P(W)\} \tag{9}$$

In the easiest case a recognizer could be made of a set of models when each represents one word. These types of recognizers usuall give the best results but they are applicable only for for very limited dictionary of words, for example the recognizer of digits. The models of words for large vocabulary task are made up from smaller units usually from phonemes (monophones models) or phonemes in context (triphone, biphone models).

A computating of likelihood of arbitrary connections set of models W has usually two parts. In the first one, the bigger units (words) are made up by connection of the models of smaller units (phonemes) according to the dictionary. And in the second one recognition

net is created from the possibly connections bigger units. This net allows a generating more possibly hypothesis which are referred by *Language model*. This process is called *Decoding* and some methods this procedure are described in section 2.2.

2.2 Decoding

The issue from decoder is to find the most likely hypothesis for an unknown utterance. The likelihoods of hypothesis are compounded from two essential parts:

- **Acoustic Model Likelihood** - likelihood generated by acoustic models (HMMs, Neural Net) which represent hypothesis.
- **Language Model Likelihood** - likelihood generated by Language model. It is based on the probabilities of words only (unigram) or more or less complex combination of ones (n-gram).

A count of hypothesis can be huge and infinite in a theoretical case thus we need efficient algorithms for pruning and rescoring field of possibly hypothesis.

2.2.1 Time Synchronous Decoding

In case of a recognizer of isolated words only (for example CAT, DOG, AND) can be created very simply net what contain three parallel branches. In the each one is a model of word (or a connection of model of phonemes which represent this word). In the start and end of net are models of silence. Output from recognizer is state sequence which give the biggest likelihood for unknown input utterance.

This net can be very simply extended for perform continuous speech recognition by adding transition from end of branches to start ones. This architecture is on picture 3.

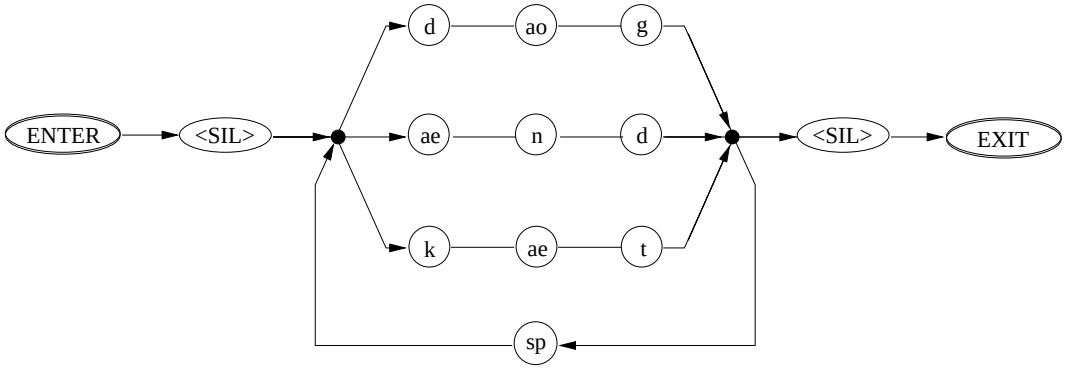

Figure 3: The simplest architecture for continuous speech recognition.

The output sentence from this decoder could be arbitrary combination of the words from dictionary but a combination of words without meaning have the same weight like combination what make a sense. For improve this problem can be implemented Language Model.

Language Model Likelihood can be calculated like:

$$P_{HW} = (Prob(W|H))^{\omega} + \rho \tag{10}$$

Where $Prob(W|H)$ is conditional probability of word W following a partial hypothesis H. ω is grammar scale factor used for adjusting range of HMM's likelihoods and Language Model probabilities. And ρ is a word insertion penalty used to control word insertion and deletion errors.

Example of extending of figure 3. by Language model is on figure 4.

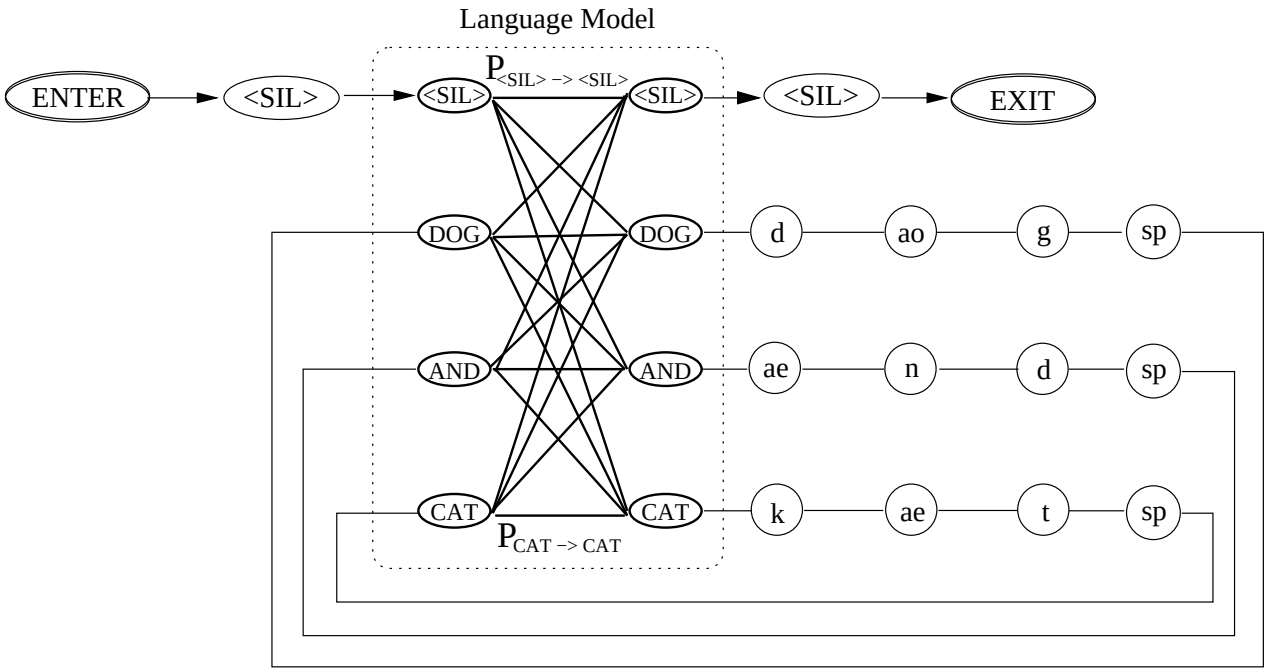

Figure 4: Example of Bigram Net.

This net is still very simply one what is using only context independent phonemes (monophones). For next increase accuracy, each phoneme can be replaced by its context dependent congener. For example, phonetic expansion word DOG (d, ao, g) will be treated like "d+ao", "d-ao+g","ao-g". Where "d+ao" mean models of phoneme "d" what have in right context phonemes "ao", "d-ao+g" models of phoneme "ao" what have in left context phoneme "d" and in right context phoneme "g" ....

This decoder reaches pretty good results but better accuracy yield **crossword external triphones** due to context dependency between words. But this improvement bring new complexity to construction of network because every first (last) phoneme in each branch must be expanded for all last (first) phonemes of words existing in dictionary.

Example of expansion this network (fig. 4.) is on the picture 5.

Basic formula for computation Language model probability is

$$Prob(W|H) = N(W,H)/N(H) \tag{11}$$

Where $N(W,H)$ is count of occurrences of a word W followed hypothesis H and $N(H)$ is count of hypothesis H.
At least, this equation (11.) covers only seeing combination of words W and hypothesis H but for practical using Language models is important to cover all combinations of ones. Probability unseeing or few occurenced n-grams could be calculated with help "back-off"

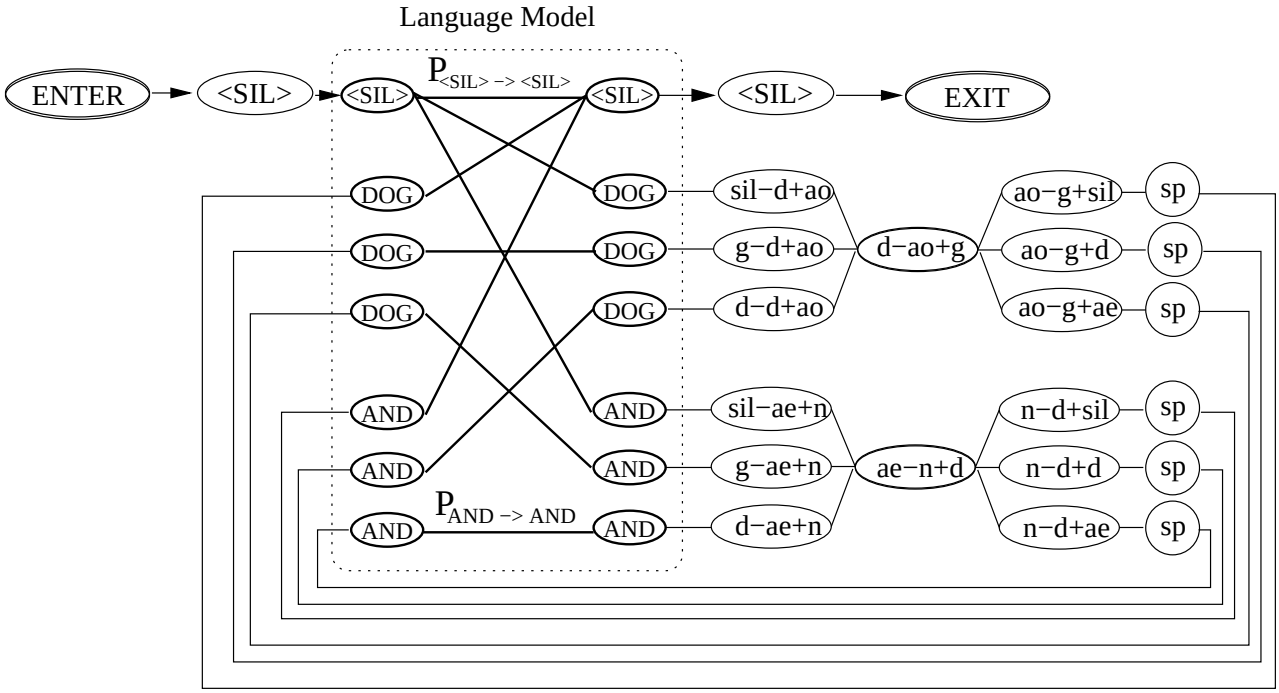

Figure 5: Crossword triphone expansion.

weight and (n-1)-gram value. Back-off weight is suitably chosen in order to ensure sum of n-gram probabilities to one. This introduce **Back-off Language models**.
Or more generally

$$Prob(W|H) = \begin{cases} (N(W,H) - D)/N(H) & \text{for} \quad N(W,H) > p \\ \\ b(H)Prob(W|H_{-1}) & \text{for} \quad N(W,H) \leq p \end{cases} \tag{12}$$

Where $b(H)$ is back-off weight of hypothesis H, $Prob(W|H_{-1})$ is conditional probability of word W following hypothesis H without first word. Consequently, if $Prob(W|H_{-1})$ is missing it could be calculated from $b(H_{-1})Prob(W|H_{-2})$ ...etc. p is threshold and D is discount value. In discount processing is small part of big n-grams count deducted and distributed among infrequently ones.

In the simplest case of bigram net the back-off weight $b(H)$ is reduced only on the weight of word before W and $Prob(W|H_{-1})$ is only unigram probability of the word W. The independency between these booth components can be used for saving some computation. In the previous cases each word end had transition to start every word. Here is possibly multiply network to two identic parts (bigram and back-off) and in the back-off one to merge all inputs from previous words to one "back-off node" after scaling by back-off weights [5]. Now it is possibly to merge same first few phonemes among words and to construct tree. Leaf nodes this tree are scaled by corresponding unigram probability.
Example of network using Back-off Language model is on picture 6.

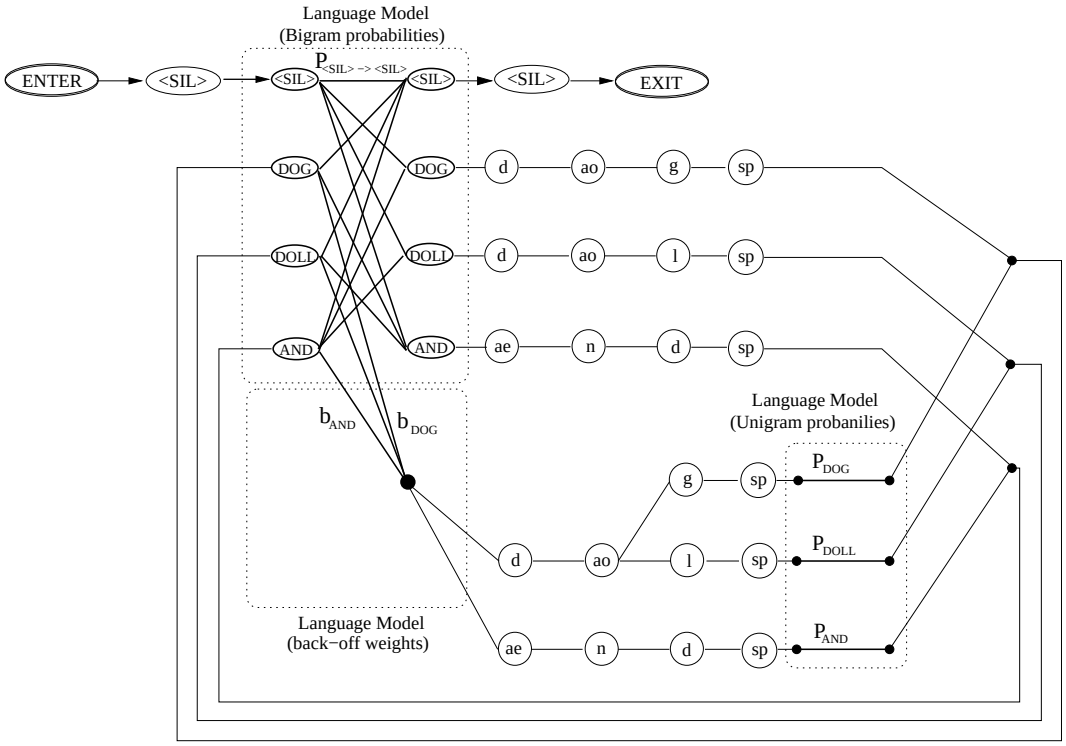

Figure 6: Example of Back-off Bigram Net.

2.2.2 Best First Decoding

In before case, time synchronous decoding, are frame by frame all hypothesis advanced without importance their likelihood. Beam search pruning [**?**] is small exemption because here is search space limited to beam in order to maximal partial likelihood and beam width. But still is a lot of computation wasted for relative unlikely path.

In the best first decoding are hypothesis sorted in order to partial likelihood and only the most likely one is updated. When this extension become unpromising a less likely hypothesis will be extended. This asynchronous approach require compare partial path with different length for finding the most probably one. Part of utterance which is not covered both of hypotheses is used for this operation. This *lookahead* could be performed by common Viterbi beam search algorithm but width of beam can be much lower.

A best first search can be implemented with A^* **search algorithm**
This method for a estimate which partial path will be extended uses "total likelihood", likelihood of whole utterance.

Total likelihood $h^*(p_q(t))$ of the partial path $p_q(t)$ is calculated by sum of two component:

- $f(p_q(t)$ - exact log likelihood calculated from start of utterance till time t.
- $g^*(p_q(t)$ - log likelihood of a estimate of completion from time t till end of utterance.

The search started with null path ($f(p_q(0)$ and $g^*(p_q(0))$ which is expanded by all words extensions. Next there are calculated exact and approximate probabilities of each words extension for selection partial path which will be expanded next.

A one kind of first best decoders are is **Stack decoders**.
Partial paths are sorted in stack in order to likelihood. The most likely path is popped from top of the stack. After there are aplyied the fast language a acoustic model matches for limiting search space for word extension of this path. Next the detail acoustic and language model matches are used for rescoring and precise selection. Updated paths are inserted back to the stack.

3 Meeting data recognition

My work is intended to recognition meeting data covered by M4 project . But, what exactly does it mean and where is basic problems?

Issues of meeting data recognition expect a continuous speech-to-text transcription of dialog from native and **non-native** speakers. Dialog is **spontaneous** conference about 4 people recorded by a field of microphones or a hands-free microphones in closed place, meeting room, component of M4 project.

From this assumption issues next problems:

- **Crosstalks** - dialog is a spontaneous speaking group of people thus crosstalks, time when minimally two participants speak at same moment, are expected. This points are totally inappropriate for speech recognition because they look like ordinary speech but it is not. Common partition by Voice Activity Detector (VAD) to speech and non-speech is inapplicable. An apportionment to speech, non-speech and crosstalk is needed.

- **Non-native speakers** - presumption a recognizing non-native speakers is essential part of M4 project but it bring some distinctions in voice. For foreign people do not good matched models trained for native speakers and there is almost impossible train models for all dialects of foreign participant. Thus, robust methods of training and good speaker adaptation is needed. Or, it open new approaches for nonstandard methods speech recognition (like TRAPS [11]) because common features (MFCC, PLP...) are very sensitive for change environment.

- **Vocal noise and Hesitance words** - In case of speed and life talk a lot of vocal noises like laugh are to be expected. And by contraries, participants could be very suspensive what spawn many hesitance words like hmm-hmm, ehm-ehm ... It is bring new interesting problems to training Language model because do not exist too much labeled data closed to meeting data.

- **Unfinished words** - In order to spontaneous talk, the meeting records contain a considerable amount of unfinished, bad pronunciated (almost from foreign people) words and a small pile of non-existed ones. This anomalities is almost impossible precise detected thus there is common to ignore them and hope that number occurrences this ones is relatively small.

4 Experiments

4.1 Data used for work

For training hidden Markov Models and testing accuracy have been used ICSI-meetings database, issue from The Meeting Recorder Project at ICSI (`http://www.ICSI.Berkeley.EDU/Speech/mr/`). This project had started at 2001 February in ICSI (International Computer Science Institute) and still is in progress. Voice from a participants of meeting had recorded by four wireless microphones and four standard computers headset microphones. Scheme of recording system is in Figure 7.

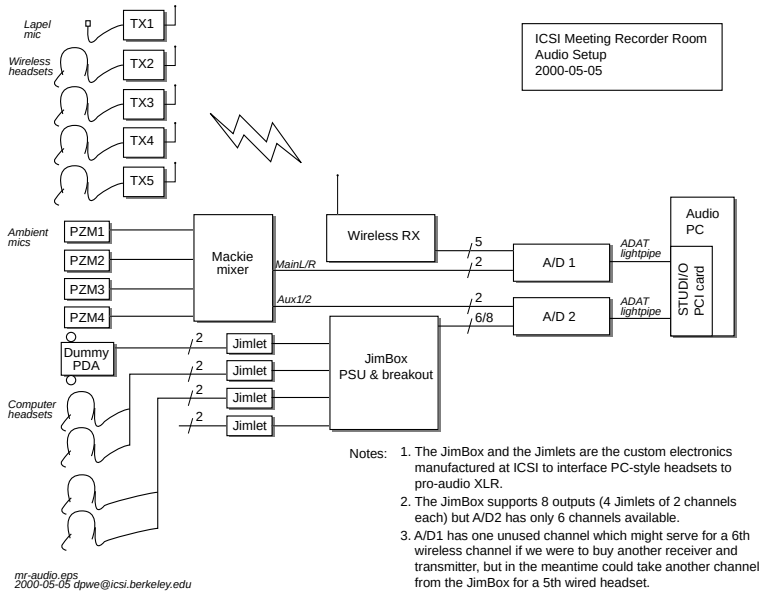

Figure 7: ICSI - Scheme of meeting recording system.

Brief description ICSI database

- **Voice format - NIST** format compressed by *shorten* encoding (standard Huffman's compression). Each meeting directory contain a one file per one microphone (all participants and this room and rooms around records).
- **Label format - XML** format. Each meeting is described in one file. Labels are in time continuance split into segments where the header of ones contains start time, stop time, and identification string of participant.
- **Number of meetings** - 46 meetings
- **Number of participants** - 27 male (15 native and 12 non-native speakers), 6 female (4 native and 2 non-native speakers)
- **Total number of hours** - 42 hours meeting time

Note: I use older release than is actual one consequently number of meetings, hours and participants is relevant only for my experiments.

This database was split into training and testing part according a kind (male,female, native and non-native) and occurrences of meeting participants. For the testing set was important to elect 1-3 meetings which contains sufficient amount of all kind of participants and these ones remove from rest of data. This election had to be very thoughtful in order to a good spreading of speakers to all meeting.

Result of this selection:

- Test part - 3 meetings (Bed13,Bed14,Bed17) in total time 2.6 hours.
- Train part - rest of the data (43meetings - 39.4 hours)

Used data for training and testing HMMs

- Training Context Independent models (CI) - 15 hours randomly chosen data from train part of database.
- Training Context Dependent models (CD) - Overall train database.
- Testing - 1 hour randomly chosen data from testing part.

Used data for experiments with Language Models

- ICSI-meetings database, training+testing part - 53099+3381 sentences.
- Switchboard database - 248581 sentences.
- BBC news radio - 1379477 sentences.
- 19 Books downloaded from Gutenberg project `http://promo.net/pg/` - 195239 sentences.
- Carefully fetched up texts from speech conference articles (ICSSLP, Eurospeech, ICSLP, NIPS) - 655886 sentences.

Size of dictionaries

- ICSI dictionary - 38617 words
- SWB dictionary - 27676 words
- Merge ICSI and SWB - 40490 words

4.2 Used tools

4.2.1 Training and testing HMMs

For work with HMMs I have used **HTK toolkit** which was written by S. Young and P.C. Woodland on Cambridge university (Great Britain) [1]. This toolkit contains a lot of tools for work with HMMs.
The most important are

- HCopy - feature extraction.

- HDict - work with dictionaries.
- HLab - work with labels.
- HComp - for initialization HMMs.
- HERest - adaptive training HMMs.
- HRest - training HMMs.
- HResult - work with results records from decoder.
- HBuild - create lattice from a bigram model.
- HParse - create lattice from net.
- HVite - Viterbi decoder.

Decoder given by HTK, HVite, is very efficient tool for common recognizing like recognition isolated words or continuous speech recognition with simply network and relative small dictionary. But for LVCSR task this one have some considerable drawbacks.

- Essentially, HVite is not optimalized for searching in dictionary which contains tens or hundreds thousand words. This is a considerable problem for experiments because recognition with a relative short test set take a lot of time.
- Support more than bigram back-off language model is not implemented. For continuous speech recognition is very important a using minimally trigram language model.

Consequently, for initial work was used HVite tool but for next experiments I have worked with **DU-coder**. Stack decoder (see in section 2.2.2) which was written in Gerhard-Mercator-University in Duisburg (Germany).

Some main attributes of this decoder can be described in some bullets:

- Based on standard stack-decoders principles but it allows skipping stack in fixed or conditional bounds or envelope search [6]. This extended skills can save some computation.
- Against HVite DU-coder allows to work with n-gram Language models. Support Language model smearing and smoothing is implemented.
- Using almost same libraries like HTK. This solve the problems with compatibly of features and models created by HTK tools.
- Propagation crossword external triphones is not implemented. This is big disadvantage of this decoder.

4.2.2 Training language models

For a training all language models was used SRILM toolkit (`http://www.speech.sri.com/projects/srilm/`) written by *SRI Speech Technology and Research Laboratory*. This toolkit consist of the tools for not only creating and evaluating statistical language models but manipulation with N-best list and word latices. Main tools are:

- ngram-count - create n-gram language models from train data or it can read test model and test data for evaluation. Kind of models created by this tool can be very variable. N-gram-count support a many type of models, smoothing and discount algorithms. Default is standard trigram language model with Good-Turing discounting and Katz back-off for smoothing [14].
- ngram-merge - merging language models. It could be used in case of insufficient memory storage.
- ngram - applying language models.
- lattice-tool - re-score and expand word lattice

4.3 Experimental results

All HMMs was trained by same setup like it was made during my first experiments with context dependent and context independent models [7]. Only initial CD models was not derived from one Gaussian models calculated by reestimation from the flatstart but from models based on alignment data by 32 Gaussian models. Because quality of tying states triphone models created by *decision tree* method [7] is influenced by precise a input one mixture models.

For all experiments was necessary to train language models which enough cover test part of database but number of sentences in the train part database (around 53000) is insufficient amount of data for making up a good language model. Very closed to ICSI meetings database is Switchboard one because this database is a spontaneous speech record too. Consequently, this database was added to language model training data and ICSI dictionary was extended by words and pronunciations from Switchboard one for next increasing training data.

4.3.1 Test for optimal features with HVite

The issues from the first experiment was to compare accuracy between PLP_E_D_A and PLP_0_D_A for next testing. The decoding run with HVite tool. According some older experiments on ICSI database was language model scaling factor approximately setted to 9.

In order to results from the tabular 1. was chosen **PLP_E_D_A** like features for next work.
Next improvement bring a transition to context dependent models especially crossword triphones.

No. of Gauss.	Accuracy with PLP_E_D_A [%]	Accuracy with PLP_0_D_A [%]
1	13.35	-
2	19.64	-
4	23.29	-
8	27.23	-
16	29.19	-
32	31.12	30.25
Note: Computation time per hour - about one week		

Table 1: Result for HVite with PLP_E_D_A and PLP_E_D_A features and context independent models.

4.3.2 Crossword external triphones with HVite

Triphone models created by decision tree method [7] which is implemented in the HHed HTK tool had this basic characteristics:

- Number of physical HMMs - 90992
- Number of logical HMMs - 26214
- Number of tying states - 7616

No. of Gauss.	Accuracy [%]
1	28.27
32	33.83
Note: Computation time per hour - about one month	

Table 2: Result for HVite with PLP_E_D_A features and context independent models.

From computation time in the experiments with HVite (Tab. 2. and 1) issues an impossibility to use HVite for next test.

4.3.3 Comparison efficiency of HVite and DU-coder

Essentially, DU-coder like each stack decoder offer much more kind of pruning than HVite because a search of the most probably sentence is more complex (see in section 2.2.2) and each part of this algorithm allows specific type of pruning. It have considerable effect on a speed of computation and accuracy.

For basic comparison was used two different settings of pruning, 32 Gaussian context independent models and PLP_E_D_A features. Other setting was same like in experiments above.

1. **Prune All**
   - Stack parameters - 12,2,0 (Stack size, skip each n-1 stack, hypothesis attraction)
   - Stack beam width - 40

- Word end beam width - 30
- Pruning beam width - 200.0 60.0
- Cross tree beam width - 220.0

2. **Only Beam search**
   - Stack parameters - 12,2,0 (Stack size, skip each n-1 stack, hypothesis attraction)
   - Stack beam width - off
   - Word end beam width - off
   - Pruning beam width - 200.0 60.0
   - Cross tree beam width - off

Type of Prune	Accuracy [%]	Time
Only Beam search	29.44	3 days
Prune All	23.05	8 hours

Table 3: Influence of pruning on efficiency of recognition with DU-coder.

How was expected, with the DU-coder in case "Prune All" setting is possibly reach some preliminary results more time earlier than with HVite (Tab. 3.).

4.3.4 Language model experiments with DU-coder

Language model based on labels ICSI meetings a Switchboard database is rude estimation appropriate configuration of data for training. Accuracy of recognition can be affected by many parameters like size of data and proportion of databases.

For experiments with a language model adjusting was used DU-coder with "Prune All" settings.

Quantity of training data					Accuracy [%]
ICSI	SWB	Speech	BBC	Books	
1	1	0	0	0	23.05
4	1	0	0	0	24.23
5	1	0	0	0	**25.02**
6	1	0	0	0	24.80
10	1	0	0	0	23.76
20	1	0	0	0	23.35
1	0	0	0	0	4.77
1	1	1	1	1	21.76
5	1	1	0	0	24.79
10	2	1	1	1	22.96

Table 4: Language models experiments.

5 Plans

This experiments are a many interesting work but in next is needed to investigate many things.

1. Test other features (MFCC, TRAPS).
2. Try to adjust next DU-coder setting (mode, size of stack, number of skipping stacks....)
3. Test of influence trigram (or four-gram) on accuracy.
4. Incorporate Switchboard database to training acoustic models.
5. Try to rescoring lattices by n-gram language models. HVite is able to generate lattices and against the DU-coder can work with crossword external triphones.

6 Conclusion

From tab 4 issues very a important and strong influence of language models. Level in accuracy of recognition is still very low. It lead

References

[1] Young S., Jansen J., Odell J., Ollanson D., Woodland P., The HTK book, Entropics Cambridge Research Lab., 1996, `http://htk.eng.cam.ac.uk`

[2] Psutka J., Komunikace s počítačem mluvenou řečí, Academica, Praha, 1995

[3] Schwarzp P., Diplomová práce,FEI VUT Brno, Brno, 2000

[4] Pálková Z., Fonetika a fonologie češtiny, Praha, Karolinum, 1994

[5] Odell J., The Use of Context in Large Vocabulary Speech Recognition,Queens' College, March 1995

[6] Willet D., Neukirchen Ch., Rigoll G., DUCODER - The Duisburg University LVCSR Stack Decoder, Department of Computer Science Faculty of Electrical Engineering Gerhard-Mercator-University Duisburg, Germany,2000

[7] Karafiát M., Černocky J., Context depended Hidden Markov Models in recognition of Czech, Faculty of Information Technology, Brno University of Technology, RadioElektronika 2002

[8] Karafiát M., Diplomová práce, FEI VUT Brno, Brno, 2001

[9] Gogineni S., Search Strategies in Speech Recognition, Institute for Signal and Information Processing Department of Electrical and Computer Engineering Missippi State University, 2000

[10] Deshmukh N., Efficient Search Algorithms For Large Vocabulary Continous Speech Recognition, Institute for Signal and Information Processing Department of Electrical and Computer Engineering Missippi State University, 1996

[11] Sangita S., Ellis D., Kajarekar S., Pratibha J., Hermansky H., Feature Extraction Using Non-Linear Transformation for Robust Speech Recognition on Aurora Database, Oregon Graduate Institute of Science and Technology, Portland, Oregon, USA, ICASSP 2000

[12] Leggetter C.J., Woodland P.C., Flexible Speaker Adaptation Using Maximum Likelihood Linear Regression, Cambridge University Engineering Department, Cambridge, UK, 1995

[13] Woodland P.C., Pye D., Gales M.J.F., Iterative Unsupervised Adaptation Using Maximum Likelihood Linear Regression, Cambridge University Engineering Department, Cambridge, UK, ICSLP 1996

[14] Wu J.,Zheng F.,On Enhancing Katz-Smoothing Based Back-Off Language Model, Center of Speech Technology Department of Computer Science and Technology, Tsinghua University, Beijing, China, ICSLP 2000