INFORMATICA, 2004, Vol. 15, No. 1, 93–110
© 2004 *Institute of Mathematics and Informatics, Vilnius*

Specifics of Hidden Markov Model Modifications for Large Vocabulary Continuous Speech Recognition

Darius ŠILINGAS
Department of Applied Informatics, Vytautas Magnus University
Vileikos 8, LT-3035 Kaunas, Lithuania
e-mail: i5dasi@vaidila.vdu.lt

Laimutis TELKSNYS
Department of Applied Informatics, Vytautas Magnus University
Recognition Processes Department, Institute of Mathematics and Informatics
Goštauto 12–205, 08663 Vilnius, Lithuania
e-mail: telksnys@ktl.mii.lt

Received: July 2003

Abstract. Specifics of hidden Markov model-based speech recognition are investigated. Influence of modeling simple and context-dependent phones, using simple Gaussian, two and three-component Gaussian mixture probability density functions for modeling feature distribution, and incorporating language model are discussed. Word recognition rates and model complexity criteria are used for evaluating suitability of these modifications for practical applications. Development of large vocabulary continuous speech recognition system using HTK toolkit and WSJCAM0 English speech corpus is described. Results of experimental investigations are presented.

Key words: large vocabulary continuous speech recognition, hidden Markov model, Viterbi recognition, beam search, context-dependent phones, Gaussian mixture, language modeling, HTK, WSJCAM0.

1. Introduction

Today we communicate with computers and other systems using keyboard, mouse, buttons, and other artificial devices, although for people the most natural way to communicate is speech. Despite a lot of research has been made in the speech recognition field, it is still easier task for people than for computers. Speech and language technologies have found many commercial applications, such as spelling checkers, dictation systems, voice control systems, voice information retrieval systems, and computer-assisted language learning. However, large vocabulary continuous speech recognition (LVCSR) systems still operate with error rates that are much too high to be used in practice. Some laboratory systems achieve nearly usable recognition rates but use very complex models, which does not allow real-time performance required for many practical applications.

LVCSR systems have lots of tunable parameters and modeling alternatives that can have greater or smaller effect to the recognition performance. In this paper we look inside the LVCSR system based on Hidden Markov Model (HMM) methods, which is a state-of-the-art technique used in modern speech recognition systems. We review specifics of HMM modeling for continuous speech recognition, build an experimental LVCSR system and investigate several HMM modifications.

2. Statement of the Problem

Since 1993, when resource management and WSJ0 speech corpora were released, there has been a lot of research in broadcast news recognition systems. These systems mainly investigated large vocabulary continuous speech recognition dedicated to dictation in noise-free environments. Many research activities were going on in Cambridge University in UK, see (Woodland, 1995) and (Robinson, 1995), LIMSI University in France, see (Gauvain, 1994) and (Gauvain, 1996), and various U.S. institutions, see (Young, 1998) and (Wegmann, 1999). Multiple frameworks were prepared for comparing performance of laboratory systems for broadcast news recognition. Thorough reviews of these activities are given in (Gauvain, 1996) and (Young, 1998). However, there are many aspects that are important for practical applications but are not covered in a concise way in the publications resulted from these research activities. Only early publications looked inside the structure of the recognition systems, while more recent publications usually report improvements of recognition rates without giving detailed description of recognizer. To our knowledge, there were no publications comparing multiple HMM modifications considering recognition rates and some measure of models complexity. This is important, since in many practical applications memory and computational resources are restricted and balance between improving recognition rates and increasing models complexity must be found so that the best possible recognition rate could be achieved with allowed memory and computational resources. Some aspects, such as context-dependency, see (Kershaw, 1996), were studied separately. However it's difficult to compare the influence of various modeling variants from publications that used different recognition systems and different data sets, which may significantly affect the results. We will build an HMM-based LVCSR system using HTK toolkit, and investigate the following issues using the same training and testing data sets from WSJCAM0 speech corpus:

- compare simple and context-dependent phone HMMs by recognition rates and models complexity;
- compare different numbers of Gaussian mixture components for modeling distribution of speech features by recognition rates and models complexity;
- estimate an impact of incorporating an appropriate language model to the speech recognition rate.

British English speech corpus WSJCAM0 and similar American English corpus WSJ0 were used for evaluating performance for broadcast news recognition in 1992–1994, see (Young, 1998) for detailed review of these activities. We do not have a goal to improve state-of-the-art recognition rate achieved for these corpora by sophisticated and complex laboratory systems. Rather we will investigate listed HMM modifications for modeling speech *acoustics* (without incorporating sophisticated language models which can improve absolute recognition rate) and use comparative analysis of their recognition rates and computational complexity necessary for such modeling.

3. Specifics of HMM-Based LVCSR

The continuous speech recognition problem is to recognize word sequence $W = W_1, \ldots, W_n$ from observed acoustic speech signal, by which speaker communicates this word sequence W. We observe a speech signal, which is coded by $Y = Y_1, \ldots, Y_T$, and our task is to recognize W analyzing observed Y. Statistically this problem is formulated as $\widehat{W} = \arg\max_W P(W|Y)$. It is not feasible directly, but applying Bayes rule we can split this problem into feasible components:

$$\widehat{W} = \arg\max_W P(W|Y) = \arg\max_W P(Y|W) \cdot P(W). \tag{1}$$

The values of $P(Y|W)$ and $P(W)$ can be estimated from parametric statistical models – respectively acoustic model and language model.

Fig. 1 shows a graphical illustration of typical LVCSR system design, which we applied in our experimental system. The first component in LVCSR system is a signal processing front-end, which converts raw speech waveform into sequence of feature vectors $Y = Y_1, \ldots, Y_T$. It is well known that the raw speech signal carries a lot of redundant information and thus is not the best representation for using in recognition systems. It is

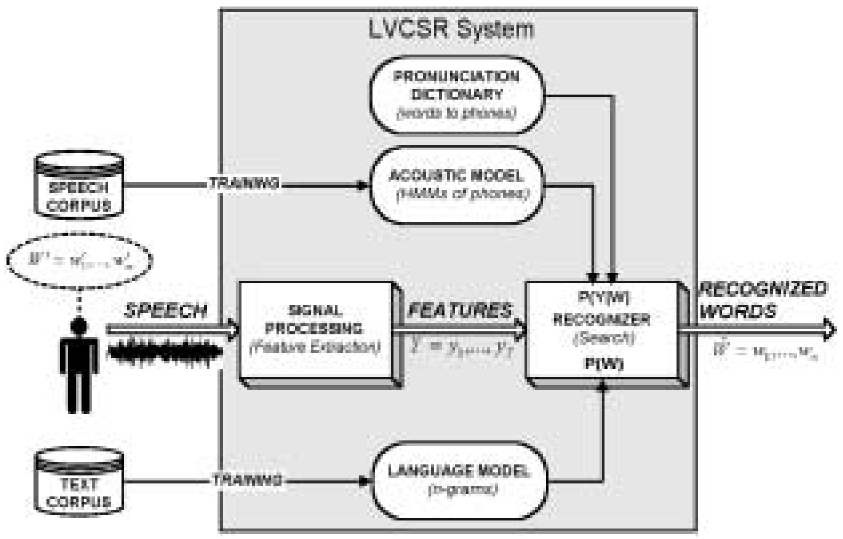


Fig. 1. Structure of LVCSR system.

a common technique to segment signal into short-time overlapping frames (we use 25 ms frames shifted by 10 ms) that are treated as pseudo-stationary, and extract representative feature vector for each frame. We will denote such feature vector y_t, where t is the index of the frame corresponding to discrete time value. Reader may refer to (Deller, 1993) for detailed discussion about extracting various feature sets.

We have used a set of 39 features, which can be divided into 3 streams:

- 13 primary feature set $c_1^t, \ldots, c_{13}^t$, consisting of 12 Mel frequency cepstral coefficients (MFCC) and logarithm of signal energy;
- 13 first order time derivatives $\Delta c_1^t, \ldots, \Delta c_{13}^t$, where $\Delta c_i^t = c_i^t - c_i^{t-1}$, $i = 1, \ldots, 13$;
- 13 second order time derivatives $\Delta^2 c_1^t, \ldots, \Delta^2 c_{13}^t$, where $\Delta^2 c_i^t = \Delta c_i^t - \Delta c_i^{t-1}$, $i = 1, \ldots, 13$.

This feature vector sequence Y is then passed to recognizer, which computes likelihood score corresponding to $P(Y|W)$ and incorporates likelihood score corresponding to $P(W)$.

The state-of-the-art technique in modeling acoustics of speech, i.e., the $P(Y|W)$ component, is Hidden Markov Model (HMM). HMM is a doubly stochastic process, consisting of observable stochastic output process and not-observable (internal) stochastic state transition process. HMM changes a state each discrete time step t, and the state generates an observable output. Probability of observing a particular output value is probabilistically dependent on the state, in which HMM is at that time. A single HMM has the following parameters:

- Number of internal states N, which is selected according to HMM specifics. We will use variable s_t for denoting number of state in that HMM is at discrete time t. We use three-state HMMs (see Fig. 2).
- The possibilities of transitions between internal states. HMM state transition process is a first-order Markov chain, i.e., $P(s_t|s_1, s_2, \ldots, s_{t-1}) = P(s_t|s_{t-1})$. Therefore transition probabilities can be defined by $N \times N$ size matrix $A = \{a_{ij}\}$, $i, j = 1, \ldots, N$, where $a_{ij} = P(s_t = j|s_{t-1} = i)$. Some transitions may be not allowed by setting their probabilities to zero. We use left-to-right phone HMMs that do not allow backward transitions (see Fig. 2).
- The functions defining observation probabilities for each state: $B = \{b_i(y)\}$, $i = 1, \ldots, N$. We use continuous HMMs with vector-valued observations, thus $b_i(y)$ will correspond to multivariate Gaussian probability distribution function (pdf),

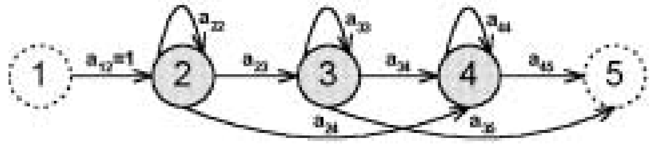


Fig. 2. Three-state left-to-right HMM topology for phones.

i.e., $b_i(y) = N(y;\mu,\Sigma)$, such that $N(y;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}}e^{-\frac{1}{2}(y-\mu')\Sigma^{-1}(y-\mu)}$ where μ is mean vector, Σ is covariance matrix, and n is the dimensionality of y. We will further extend the system to use Gaussian mixtures: $b_i(y) = \sum_{i=1}^{k} c_i \cdot N(y;\mu_i,\Sigma_i)$, where $\sum_{i=1}^{k} c_i = 1$. We will perform experiments to investigate the influence of modeling feature distribution functions with simple Gaussian and two and three-component Gaussian mixture pdf. We use diagonal covariance matrices Σ, i.e., variances only, because using full covariance matrices heavily increases computational complexity and does not give significant recognition improvement.

We use a compact notation of an HMM model $\lambda = \{A, B\}$. We think of a single HMM as a model of system generating features of modeled speech unit, which can be a word, syllable, simple or context-dependent phone. The choice of modeled speech unit depends on speech recognition system characteristics, available training data sets, and specifics of language. We will compare modeling simple and context-dependent phones in our experiment. We will call simple phones *monophones*, and context-dependent phones *triphones*. We need to define the topology of HMMs. Choosing number of states is based on vocal tract behavior. When uttering a phone, a vocal tract goes through three basic states – changing from previous phone, steady pronunciation of phone, and changing to the next phone. Also, since speech acoustics is of forward-in-time nature, we do not need backward transitions. Finally, sometimes phones are pronounced very fast and some state may be skipped. For all these reasons, all phones will be modeled by three-state left-to-right HMMs.

A topology of left-to-right HMM for any phone is shown in Fig. 2. The states 1 and 5 are shown differently because they are non-emitting and are only used for joining multiple phone HMMs together. Dealing with continuous read speech, we also need to model short pauses and silence. We model pause by one-state HMM and silence by three-state HMM, which central state shares the same pdf with the single state of pause HMM, and left and right states have forward and backward transitions to each other to model noises that may be similar to start of speech.

Having selected modeled speech units and their topologies, we may build prototype HMMs, which need to be trained using appropriate amount of speech data to represent the features of associated phones. We used the *embedded Baum–Welch* training algorithm, which is implemented in HTK, see (Young, 2002). Discussing HMM training algorithms is out of scope for this article. Reader may refer to (Rabiner, 1993) for details of training algorithms.

Recognizing with trained HMMs, we need to find HMM which produces the best likelihood score for observed speech signal. There are two approaches to evaluating HMM likelihood scores used for recognition: evaluate accumulated *any path* likelihood score or evaluate *best path* likelihood score. We will take the *best path* approach and use *Viterbi* algorithm, which finds the best state sequence and computes its likelihood score. Let's define variable $\delta_t(i) = \max_{s_1,s_2,\ldots,s_t} P(s_1,s_2,\ldots,s_t = i; y_1,y_2,\ldots,y_t|\lambda)$, which indicates the best likelihood score till discrete time moment t for state sequence ending in the state i. By induction, $\delta_{t+1}(j) = \max_i(\delta_t(i)\cdot a_{ij})\cdot b_j(y_{t+1})$. To avoid floating-point

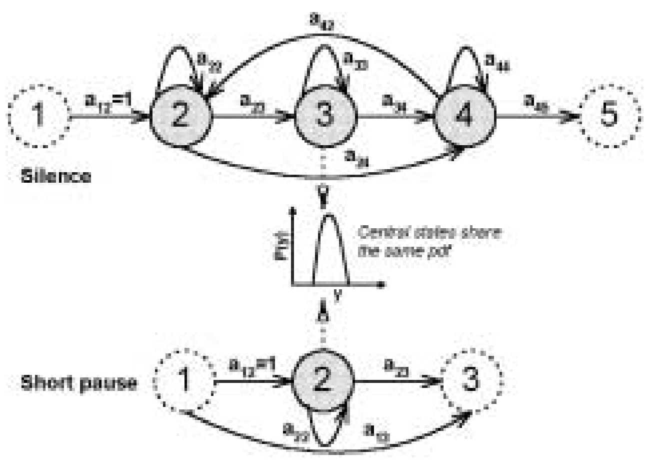


Fig. 3. Silence and short pause HMM topologies.

numbers underflow problems we apply negative logarithm, which turns likelihood scores into cost scores and maximization into minimization. The full Viterbi algorithm in logarithm form is given bellow.

First, we need to initialize $\delta_1(i)$ values:

$$\delta_1(1) = -\log_{10}(b_1(y_1)), \tag{2}$$

$$\delta_1(i) = \infty, \ \ i = 2, \ldots, N. \tag{3}$$

Then we can use induction for finding forward-in-time $\delta_t(j)$ values:

$$\delta_t(j) = \min_{1\leqslant i\leqslant N}(-\log_{10}(\delta_{t-1}(i))) - \log_{10}(b_j(y_t)), \quad t=2,\ldots,T;\ j=1,\ldots,N. \tag{4}$$

To backtrack the best state sequence, we need to store arguments that minimize $\delta_t(j)$ for each t and j:

$$\psi_t(j) = \arg\min_{1\leqslant i\leqslant N}\bigl(-\log_{10}(\delta_{t-1}(i)) - \log_{10}(a_{ij})\bigr), \quad t=2,\ldots,T;\ j=1,\ldots,N. \tag{5}$$

Finally, the minimum cost score can be found by:

$$P^* = \min_{1\leqslant i\leqslant N}(\delta_T(i)). \tag{6}$$

For backtracking the best state sequence we use:

$$s_T^* = \arg\min_{1\leqslant i\leqslant N}(\delta_T(i)), \tag{7}$$

$$s_t^* = \psi_{t+1}(s_{t+1}^*), \quad t = T-1, \ldots, 1. \tag{8}$$

The complexity of Viterbi algorithm is $O(N^2T)$, which is practically feasible. However, for large vocabulary, when we need to concatenate a sequence of phone HMMs to form words and sequences, the search space gets too large. A *beam search* allows dropping unlikely hypotheses, which may drastically reduce computations without having significant impact on results. This simple semi-greedy approach is known to work well in practice. It works within Viterbi search algorithm as follows:

At each time step t, compute minimum cost score for previous step $t-1$:

$$P_{t-1}^* = \min_{1 \leqslant i \leqslant N} (\delta_{t-1}(i)). \tag{9}$$

Using P_{t-1}^* and carefully chosen constant K (we used $K = 300$), define a dynamic threshold:

$$\tau_{t-1} = K \cdot P_{t-1}^*. \tag{10}$$

Then purge, i.e., do not consider in minimization (5)*, the states at level $t-1$, for which cost score is higher than the threshold: $\delta_{t-1}(i) > \tau_{t-1}$.*

As already mentioned, recognizing continuous speech uses joining phone HMMs into sequences to form words, and joining words into sequences to form sentences. Possible sequences of HMMs are defined by pronunciation dictionary, which contains all allowed words and their phonetic transcriptions. Since we will model monophones and triphones, we need a monophone and triphone pronunciation dictionaries.

Table 1 shows several words from vocabulary used in our experiment and their monophone and triphone pronunciation transcriptions. Some words have multiple pronunciations. Note that when using monophones phone *ax* would be modeled by the same HMM in all four words, and using triphones there would be different triphone HMMs for this phone in all four words.

For continuous speech recognition we need to incorporate the linguistic knowledge – the $P(W)$ score. We allow arbitrary word sequences with likelihood scores given

Table 1
Sample monophone and triphone pronunciation transcriptions

Word	*Monophone transcription*	*Triphone transcription*
OUR	awl ax	awl+ax awl-ax
OUR	awl ax r	awl+ax awl-ax+r ax-r
OURS	awl ax z	awl+ax awl-ax+z ax-z
OURSELF	awl ax s eh l f	awl+ax awl-ax+s ax-s+eh s-eh+l eh-l+f l-f

by context-free stochastic language models. The most widely used statistical context-free technique is n-*grams*, which assumes Markov chain idea that probability of observing a word is only dependent on the preceding n words: $P(w_k|w_{k-1},\dots,w_1) \approx P(w_k|w_{k-1},\dots,w_{k-n-1})$. We have used *bigram* ($N=2$) language model. Reader may refer to (Clarkson, 1997) for more detailed explanation of n-gram modeling algorithms. The basic formula for computing unigram and bigram probabilistic scores is:

$$P(w_i) = \frac{K(w_i)}{K}, \quad i = 1,\dots,L, \tag{11}$$

$$P(w_i|w_j) = \begin{cases} \dfrac{K(w_jw_i) - D}{K(w_j)} & \text{if } K(w_jw_i > k), \\ b(w_j)\cdot P(w_i) & \text{else,} \end{cases} \qquad i,j = 1,\dots,L. \tag{12}$$

Unigram score $P(w_i)$ is an estimated probability of observing the ith word from the vocabulary calculated using simple relative frequencies. $K(w_i)$ is the number of word w_i observations. K is the total number of words in text corpus. L is the number of words in vocabulary. Bigram score $P(w_i|w_j)$ is an estimate of conditional probability of observing word w_i if previous word was w_j. $K(w_jw_i)$ is the number of word pairs w_jw_i observed in text corpus. Since there are L^2 possible word pairs, we need to do some language model smoothing, i.e., introduce some discount for frequent observations and assign some small probabilities for pairs that are rarely or never found in text corpus. This is implemented by introducing coefficients D and $b(w_j)$. Reader may refer to (Clarkson, 1997) for a short introduction to computing n-grams.

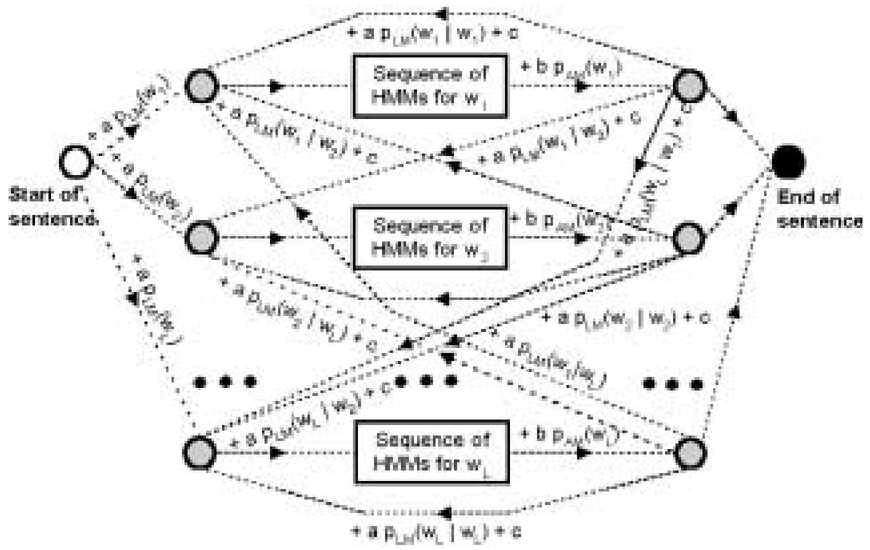


Fig. 4. The token passing HMM search lattice incorporating linguistic scores.

The actual search algorithm that we used is alternative *Viterbi search* formulation called *token passing*, see (Young, 1989). Fig. 4 contains graphical illustration of the token passing search algorithm. Searching for the best sequence is implemented by passing a token, which accumulates scores from language model (LM) and acoustic model (AM). We need to incorporate scores from language model in the search algorithm as well. Since we applied negative logarithm to acoustics likelihood score, we need to do the same with linguistic score: $p_{LM}(w_i) = -\log_{10} P(w_i)$ and $p_{LM}(w_i|w_j) = -\log_{10} P(w_i|w_j)$. To balance the impacts of scores from acoustic models p_{AM}, and linguistic models $p_{LM}(w_i)$, $p_{LM}(w_i|w_j)$, we add weight coefficients: a for scaling linguistic scores and b for scaling acoustic scores. The coefficient c is another means for tuning the speech recognizer – *word transition penalty*. It is known that speech recognizer may tend to recognize sequences of short words, thus word transition penalty may be helpful for compensating this effect.

We have discussed specifics of HMM modeling techniques used for continuous speech recognition in our case. We will now discuss the experimental investigations, which make practical use of described models and algorithms.

4. Experimental Investigations

We implemented an experimental large vocabulary continuous speech recognition system using the discussed theoretical methods. For this task we used HTK toolkit, which is developed by Cambridge University and is freely available at `http://htk.eng.cam.ac.uk` for non-commercial usage. HTK provides powerful and flexible tools for building this kind of system: extracting features from sound files (HCopy), manipulating label transcriptions (HLEd), training HMMs (HCompV, HERest or HInit, HRest), editing HMM structure (HHEd), realigning label transcriptions for multiple pronunciations (HVite), manipulating vocabularies (HDMan), building word networks (HBuild), computing bigram language model (HLStats), adapting HMMs for specific speaker (HEAdapt), performing recognition (HVite), and evaluating recognition results (HResults). Reader may refer to (Young, 2002) for HTK tutorial and detailed description of each HTK tool. Additionally, we used various Linux utilities, such as gawk, sort, unique, and tr, for preparing data resources.

We need a large amount of speech samples for training and testing. We have used WSJCAM0 corpus that contains sentences from *Wall Street Journal* read by British English speakers in a quiet room isolated from environmental noises. Detailed description of the WSJCAM0 corpus can be found in (Fransen, 1994). Results of pilot experiments using WSJCAM0 are reported in (Robinson, 1995). WSJCAM0 corpus contains six CDs with speech waveform files and their label transcriptions (word-level and monophone-level). Each waveform contains one sentence. WSJCAM0 defines data sets for training, development test and evaluation test for $5k$ and $20k$ word vocabularies, which are also given (without phonetic transcriptions). In our experiments we used only $5k$-word vocabulary test data set. There were 7861 sentences (132465 words, approximately 12 hours

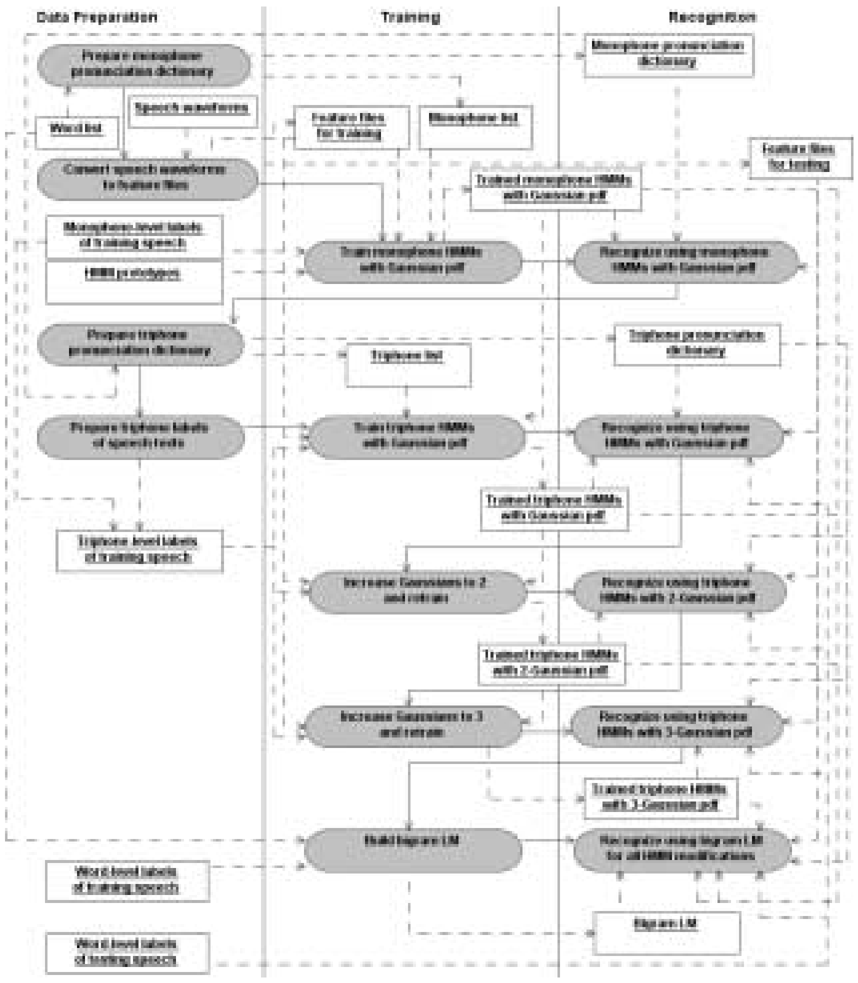

Data Preparation
Training
Recognition
Prepare monophone pronunciation dictionary
Monophone pronunciation dictionary
Word list
Speech waveforms
Feature files for training
Monophone list
Feature files for testing
Convert speech waveforms to feature files
Trained monophone HMMs with Gaussian pdf
Monophone-level labels of training speech
HMM prototypes
Train monophone HMMs with Gaussian pdf
Recognize using monophone HMMs with Gaussian pdf
Prepare triphone pronunciation dictionary
Triphone pronunciation dictionary
Triphone list
Prepare triphone labels of speech texts
Train triphone HMMs with Gaussian pdf
Recognize using triphone HMMs with Gaussian pdf
Trained triphone HMMs with Gaussian pdf
Triphone-level labels of training speech
Increase Gaussians to 2 and retrain
Recognize using triphone HMMs with 2-Gaussian pdf
Trained triphone HMMs with 2-Gaussian pdf
Increase Gaussians to 3 and retrain
Recognize using triphone HMMs with 3-Gaussian pdf
Trained triphone HMMs with 3-Gaussian pdf
Build bigram LM
Recognize using bigram LM for all HMM modifications
Word-level labels of training speech
Bigram LM
Word-level labels of testing speech

Fig. 5. Experiment workflow (UML activity diagram).

of speech) dedicated to training HMMs, and 368 sentences (6254 words) dedicated to evaluation test for $5k$-word vocabulary recognition task.

Fig. 5 shows the experiment workflow using UML activity diagram. The rounded grey-filled blocks are activities and the square blocks represent incoming and outgoing data objects. We will shortly describe each activity in a sequential order.

Since WSJCAM0 corpus contains list of words for $5k$-word vocabulary test and does not define their phonetic transcriptions, we had to prepare vocabulary with phonetic transcriptions. The activity *Prepare monophone pronunciation dictionary* takes input data *Word list* (words included in WSJCAM0 $5k$-word task vocabulary) and outputs *Monophone pronunciation dictionary*. We used BEEP dictionary, automatic phonetic transcription tool addttp4, and several manual transcriptions for building the full pronunciation dictionary. We defined pronunciations using the extended ARPAbet phone set given in

(Fransen, 1994).

For speech recognition we use speech feature vectors extracted from raw speech waveforms. We extracted features for both training and evaluation test files at once. Therefore, activity *Convert speech waveforms to feature files* takes as input *Speech waveforms* (both training and testing) and outputs *Feature files for training* and *Feature files for testing*. We used HTK tool HCopy for coding speech waveforms by feature vectors.

We created prototypes for monophone HMMs and silence and short pause. Using *HMM prototypes*, *List of monophones* (extracted from vocabulary using HDMan tool), *Mohopnohe-level labels of training speech* (provided by corpus), and *Feature files for training*, activity *Train monophone HMMs with Gaussian pdf* performs training of prototype models and outputs *Trained monophone HMMs with Gaussian pdf*, which contain HMM parameters (state transition possibilities and pdf parameters – means and variances – for each emitting state) for each element in the *Monophone List*. We used flat-start training implemented by HCompV function, which computes global speech feature means and variances that are later assigned for each HMM. Then we performed four iterations of embedded Baum–Welch training using HERest function, tied silence and short-pause HMMs using HHEd function, called two iterations of training, realigned transcriptions for multiple pronunciations using HVite function, and again performed two iterations of training.

Using *Trained monophone HMMs with Gaussian pdf*, *Monophone pronunciation dictionary*, *Feature files for testing*, and original *Word-level labels of testing speech*, activity *Recognize using monophone HMMs with Gaussian pdf* evaluates the word recognition rate of trained monophone HMMs. We used HVite function, which executes Viterbi beam-search and creates label files with recognized words for feature files. Using original and recognized word-level labels we can evaluate the performance using function HResults. We will use word recognition rates as performance measure. HResults counts total word number (N), word hits (H), word deletions (D), submitted word (S), and inserted words (I). There are two word recognition rate measures – word correctness $C = \frac{H}{N} \times 100\%$, and word accuracy $A = \frac{N-S-D-I}{N} \times 100\%$. The same recognition process is used by all succeeding activities that evaluate recognition performance for other HMM modifications.

Monophone HMMs serve as a starting point for triphone HMMs. Before moving to triphones, we need to convert vocabulary and label files to include triphone transcriptions instead of monophones. The activity *Prepare triphone pronunciation dictionary* takes *Monophone pronunciation dictionary* and converts it to *Triphone pronunciation dictionary*. It also outputs a *Triphone list*, which includes all triphones found in vocabulary. The conversion of monophone vocabulary to triphone vocabulary and creation of triphone list was performed by HTK function HDMan. The activity *Prepare triphone-level labels of training speech* takes *Monophone transcriptions of speech texts* as an input object and outputs *Triphone-level labels of training speech*. This is performed by HTK function HLEd.

Now we move to triphone HMMs that are built from trained monophone HMMs. Using *Triphone list*, *Trained monophone HMMs with Gaussian pdf*, *Feature files for* *training*, and *Triphone-level labels of training speech* as input data, activity *Train triphone HMMs with Gaussian pdf* outputs *Trained triphone HMMs with Gaussian pdf*. There are several issues in performing this activity that need to be mentioned. First, we use monophone HMM as a starting point for each biphone and triphone having that monophone as a central phone. Second, values of transition are little dependent of phone context – the greatest impact of context is in feature distribution. Therefore a single state transition matrix is shared by all monophones, biphones and triphones with the same central phone. Third, the feature distribution for central state in triphone HMM corresponds to steady phone sound. Therefore it is also shared by all monophones, biphones and triphones with the same central phone. Fourth, some triphones are found very rarely in training data, thus training their HMMs would result in poor estimates. To overcome this problem, we use triphone clustering based on question trees as suggested in (Young, 1994). For transition and state tying, triphone clustering we used HTK tool HHEd and trained triphone HMMs using three iterations of embedded Baum–Welch training. First modification of triphone HMMs and monophone HMMs model feature distribution by a Gaussian pdf. Later we will move to mixtures of two and three Gaussian components, which captures better the inter-speaker variations.

The activity *Recognize using triphone HMMs with Gaussian pdf* evaluates the word recognition rate for our *Trained triphone HMMs with Gaussian pdf*, using *Vocabulary with triphone transcriptions*, *Feature files for testing*, and *Word-level labels of testing speech*.

As we have already noted, mixture of multiple Gaussian components for feature distribution models better the inter-speaker variations, thus we will increase Gaussian components from one (simple Gaussian pdf) to two. The activity *Increase Gaussians to 2 and retrain* takes as input *Trained triphone HMMs with Gaussian pdf, Feature files for training*, and *Triphone transcriptions for speech texts* and outputs *Trained triphone HMMs with Gaussian pdf*. This activity consists of splitting Gaussian distribution into two Gaussian components and retraining HMMs. Increasing number of Gaussian components is implemented using HTK tool HHEd. It takes the Gaussian mixture component with heaviest weight and splits it into two components. Then we perform two iterations of training using HERest.

The activity *Recognize using triphone HMMs with 2-Gaussian pdf* evaluates the word recognition rate for our *Trained triphone HMMs with 2-Gaussian pdf, Vocabulary with triphone transcriptions, Feature files for testing*, and *Word-level labels of testing speech*.

The activity *Increase Gaussians to 3 and retrain* takes as input *Trained triphone HMMs with 2-Gaussian pdf, Feature files for training*, and *Triphone transcriptions for speech texts* and outputs *Trained triphone HMMs with 3-Gaussian pdf*. This is a further sophistication of Gaussian mixtures, which is performed similarly to changing from Gaussian pdf to two-component Gaussian mixture.

The activity *Recognize using triphone HMMs with 3-Gaussian pdf* evaluates the word recognition rate for our *Trained triphone HMMs with 3-Gaussian pdf, Vocabulary with triphone transcriptions, Feature files for testing*, and *Word-level labels of testing speech*.

We also need to incorporate language model (LM) scores in recognition search. Recognition search algorithm is highly dependent on used language model. HTK tool

HVite implements search algorithm, which allows only unigrams and bigrams. The activity *Build bigram LM* takes *Word-level labels of training speech* and *Word list* as input and outputs *Bigram LM*, containing unigram and bigram probabilities for each word in vocabulary. The WSJCAM0 corpus includes neither pre-computed language models nor enough text for building appropriate language models. We have used transcriptions of speech dedicated to training containing about $100k$ words. Testing sentences are not included in training data. For computing appropriate bigram LM, text amount in order of $10M$ words is necessary, thus our bigram LM is a poor estimate. To see how far we can get with appropriate bigram LM, we also computed *unfair* bigram LM, using transcriptions of both training and testing sentences, which means that probabilities of test word sequences may get unreasonably high values because of small text corpus size. We computed bigrams using HTK tool HLStats.

We evaluated recognition using *Bigram LM* for all modifications of HMMs. Additionally we evaluated recognition using *unfair* bigram LM to see how far we could get with bigram LM computed from appropriate amount of text.

Finally, we want to notice that recognition performance, using our LVCSR system built with HTK, was much slower than real-time. The recognition of about 30 minutes of speech dedicated to evaluation test, took about 7–8 hours on computer with 1.5GHz Pentium CPU and 256MB RAM. This confirms our argument that when building real-time LVCSR systems we have to give up those sophistications that increase computations heavily, but do not give significant recognition improvement.

5. Results of Experimental Investigations

We have evaluated several variants of trained HMMs on WSJCAM0 evaluation test data. We present recognition results using comparative charts and tables reporting both word recognition correctness and accuracy percentage measures.

Fig. 6 shows recognition progress going from simple to more sophisticated HMMs. Since models are getting more sophisticated, computation and memory resources increase heavily as well. The computational complexity of monophone HMMs and triphone HMMs is the same – just the number of models increases from 46 monophone models to 8694 triphone models (some of them share the same parameters). Thus this only affects the memory resources and amount of necessary training data. Changing simple Gaussian pdf with Gaussian mixtures is a different issue – it does not change the number of HMMs but increases the number of HMM parameters, which affects both computational and memory resources.

Table 2 compares complexity of triphone HMM models based on number of parameters. It is necessary to indicate which improvements are significant and which can be neglected when building practical applications. Moving from monophones to triphones makes a significant progress in recognition – 88% relatively higher correctness and 85% higher accuracy[1]. Splitting Gaussian pdf into Gaussian mixture of two and three com-

[1]When comparing recognition performance of two HMM modifications we always use relative measures: $\frac{C_2-C_1}{C_1} \times 100\%$ and $\frac{A_2-A_1}{A_1} \times 100\%$

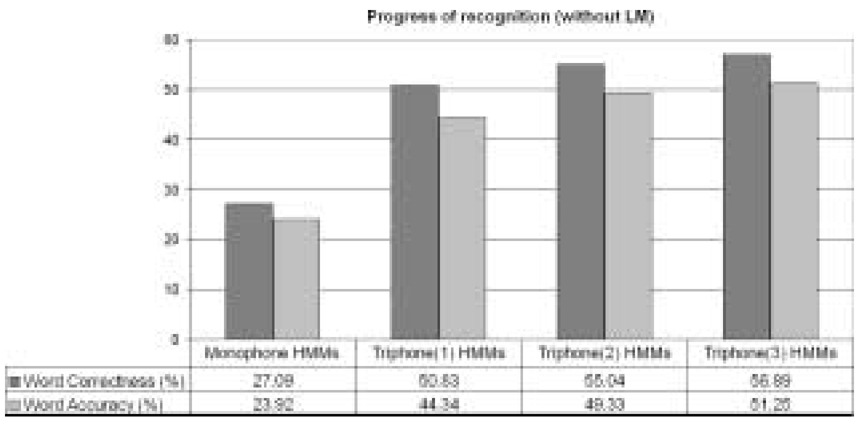
Table 2

Number of parameters in HMMs with Gaussian, 2-component Gaussian mixture, and 3-component Gaussian mixture pdf

	HMM with Gaussian pdf	*HMM with 2-component Gaussian mixture*	*HMM with 3-component Gaussian mixture*
Transition probabilities	8	8	8
Pdf parameters:			
Means and variances	(39 + 39)	((39 + 39)	((39 + 39)
Gaussian components	x 1	x 2	x 3
Additional weight coeffs		+ 2)	+ 3)
Number of states	x 3	x 3	x 3
Total parameters	242	482	719

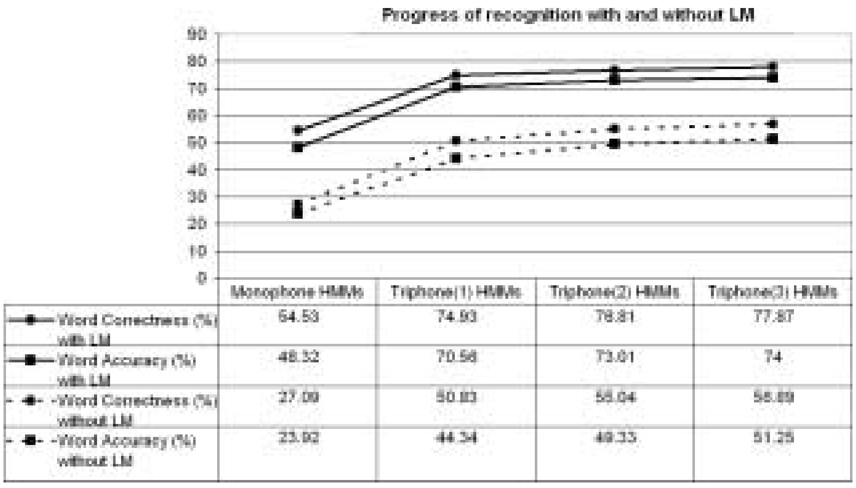
ponents improves recognition less significantly – 8% higher correctness and 11% higher accuracy comparing HMMs with two-component Gaussian mixture pdf to HMMs with Gaussian pdf, and 3% higher correctness and 4% higher accuracy comparing HMMs with three-component Gaussian mixture pdf to HMMs with two-component Gaussian mixture pdf. This suggests that modeling triphones is essential, while using multiple Gaussian component mixtures can be neglected in practical applications for performance reasons.

Fig. 7 displays comparative progress of recognition using various HMMs with and without using bigram LM. Incorporating bigram LM adds about 20%–25% of recognition correctness and accuracy. This is 101% higher correctness and 102% higher accuracy for monophone HMMs, 47% higher correctness and 59% higher accuracy for triphone HMMs with Gaussians, 40% higher correctness and 48% higher accuracy for triphone HMMs with 2-components mixtures, 37% higher correctness and 44% higher accuracy for triphone HMMs with 3-component mixtures.



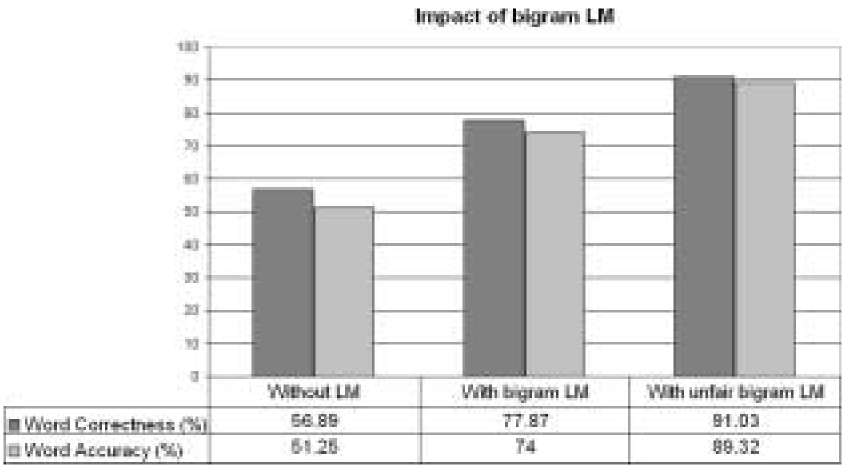
	Monophone HMMs	Triphone(1) HMMs	Triphone(2) HMMs	Triphone(3) HMMs
■ Word Correctness (%)	27.09	50.83	55.04	56.89
▨ Word Accuracy (%)	23.92	44.34	49.33	51.26

Fig. 6. Comparison of word recognition correctness and accuracy for monophone HMMs and triphone HMMs with 1, 2, and 3 Gaussian components (without LM).



	Monophone HMMs	Triphone(1) HMMs	Triphone(2) HMMs	Triphone(3) HMMs
—●— Word Correctness (%) with LM	54.53	74.93	76.81	77.87
—■— Word Accuracy (%) with LM	48.32	70.56	73.01	74
- ● - Word Correctness (%) without LM	27.09	50.83	55.04	56.89
- ■ - Word Accuracy (%) without LM	23.92	44.34	49.33	51.25

Fig. 7. Comparison of word recognition using monophone HMMs and triphone HMMs with 1, 2, and 3 Gaussian components with and without bigram LM.



	Without LM	With bigram LM	With unfair bigram LM
▩ Word Correctness (%)	56.89	77.87	91.03
▨ Word Accuracy (%)	51.25	74	89.32

Fig. 8. Comparison of word recognition using monophone HMMs and triphone HMMs with 1, 2, and 3 Gaussian components with and without bigram LM.

Fig. 8 shows another view of bigram LM impact on word recognition rates: compares using no LM, using fair (but underestimated) bigram LM, and using unfair (underestimated with artificially increased probabilities of test word sequences) bigram LM for triphone HMMs with three-component mixture pdf having the best recognition performance without LM. Using unfair LM gives 91.03% correctness and 89.32% accuracy. Comparatively, using fair (but underestimated) bigram LM shows 37% higher correctness and 44% higher accuracy as compared to not using LM, and unfair bigram LM shows 60% higher correctness and 74% higher accuracy, which could be close to the values that we could obtain with bigram LM computed using appropriate amount of text.

Finally, we want to compare our recognition rates with those reported by other researchers for the same WSJ $5k$ vocabulary task. (Robinson, 1995) reports 86.5% word recognition correctness for the same $5k$ vocabulary task achieved in pilot experiments for WSJCAM0 corpus using backed-off bigram language model computed from 38M word corpora. Review paper (Young, 1998) states that the state-of-the-art recognition rate for WSJ $5k$ vocabulary task is about 5% word error rate (WER), which corresponds to 95% word recognition accuracy. Another review paper (Gauvain, 1996) indicates that laboratory systems tested for Nov92 WSJ $5k$ vocabulary task with bigram language model achieved 6.9%–15.0% WER, Nov93 WSJ $5k$ vocabulary task with bigram language model – 8.7%–17.7%, and Nov93 WSJ $5k$ vocabulary task with *trigram* language model – 4.9%–9.2%. Although we didn't have a goal to improve these recognition rates achieved by sophisticated laboratory systems using more complex language models (5% WER is only achieved using trigrams) computed from large text corpora (38M words and more compared to our $100k$ words), our recognition rates are reasonably close to them.

6. Conclusions

Complex HMM modeling usually improves recognition rates, but also heavily increases computation and memory size, which is not tractable in most practical applications. From experimental investigations we conclude:

- Using context-dependent phones is highly preferable to simple phones – we observed about 80% relative improvement of word recognition without affecting computational complexity. Only the number of HMMs increased from 45 to 8694, which requires memory resources.
- Using two-component Gaussian mixture pdf instead of simple Gaussian pdf for modeling HMM state feature distributions is less effective: we observed only 8% relative improvement of word recognition, while the number of HMM parameters is doubled affecting both memory and computational resources. Using three-component Gaussian mixture pdf is even less effective. We suggest use mixtures only when computational and memory resources are not restricted.
- Rather that using Gaussian mixtures, incorporating appropriate language model may be explored, since even poor bigram model estimated from only about $100k$ words text, gave a significant 30%–50% relative improvement in word recognition.

7. Acknowledgements

Most of experiments with HTK and WSJCAM0 were executed during the summer school *Speech Recognition* at Magdeburg, Germany in summer of 2002. We would like to thank the staff of Recognition Systems Department at Magdeburg Otto-von-Guericke University – prof. Andreas Wendemuth, Marcel Katz, and dr. Sven Krueger – for their valuable help and the ability to use their computer environment for running experiments.

References

Clarkson, P.R., and R. Rosenfeld (1997). Statistical language modeling using the CMU Cambridge toolkit. In *Proceedings EuroSpeech'97*, Rhodes, Greece.
`http://svr-www.eng.cam.ac.uk/~prc14/eurospeech97.ps`

Cole, R., J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, G.B. Varile and A. Zampolli (1998). *Survey of the State-of-the-Art in Human Language Technology*. Cambridge University Press, Cambridge, UK.
`http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html`

Deller, J.H., J.H.L. Hansen, and J.G. Proakis (1993). *Discrete-Time Processing of Speech Signals*. Macmillan, New York, USA.

Fisher, W.H. (1999). A statistical text-to-phone function using n-grams and rules. In *Proceedings IEEE ICASSP'99*, vol.2. Phoenix, USA. pp. 649–652.
`http://www.nist.gov/speech/publications/papersrc/ttp4v11.pdf`

Fransen, J., D. Pye, T. Robinson, Ph. Woodland and S. Young (1994). *WSJCAM0 Corpus and Recording Description*. Technical report CUED/F-INFENG/TR.192, CUED Speech Group, Cambridge, UK.
`http://www.ldc.upenn.edu/Catalog/docs/wsjcam0/wsjcam0.ps`

Gauvain, J.L., L. Lamel, G. Adda and M. Adda-Decker (1994). The LIMSI Nov93 WSJ System. In *Proceedings 1994 ARPA Speech and Natural Language Workshop*, Adelaide, Australia.
`ftp://tlp.limsi.fr/public/slt94.ps.Z`

Gauvain, J.L., and L. Lamel (1996). Large vocabulary continuous speech recognition: from laboratory systems towards real-world applications. In *Transactions of the IEICE*, vol. J79-D-II N12.
`ftp://tlp.limsi.fr/public/ieice96.ps.Z`

Jurafsky, D., J.H. Martin and K.V. Linden (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, New Jersey, USA.

Kershaw, D.J. (1996). *Phonetic Context-Dependency in a Hybrid ANN/HMM Speech Recognition System*. PhD thesis, CUED, Cambridge, UK.

Rabiner, L., and B-H. Juang (1993). *Fundamentals of Speech Recognition*. Prentice Hall, New Jersey, USA.

Robinson, T., J. Fransen, D. Pye, J. Foote and S. Renals (1995). WSJCAM0: a british english speech corpus for large vocabulary continuous speech recognition. In *Proceedings IEEE ICASSP'95*, Detroit, USA. pp. 81–84.
`http://www.ldc.upenn.edu/Catalog/docs/wsjcam0/abstract.ps`

Wegmann, S., F. Scattone, I. Carp, L. Gillick, R. Roth and J. Yamron (1999). Dragon systems' 1997 broadcast news transcription system. In *Proceedings 1999 DARPA Broadcast News Workshop*, Washington, USA.
`http://www.nist.gov/speech/publications/darpa98/pdf/eng150.pdf`

Woodland, P.C., C.J. Leggetter, J.J. Odell, V. Valtchev and S.J. Young (1995). The development of the 1994 HTK large vocabulary speech recognition system. In *Proceedings 1995 ARPA Spoken Language Technology Workshop*, Austin, TX, USA.
`ftp://svr-ftp.eng.cam.ac.uk/pub/reports/woodland_slt95.ps.gz`

Young, S., N.H. Russell and J.H.S Thornton (1989). *Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems*. Technical report CUED/F-INFENG/TR.38, CUED, Cambridge, UK.
`ftp://svr-ftp.eng.cam.ac.uk/pub/reports/young_tr38.ps.Z`

Young, S., J. Odell and Ph. Woodland (1994). Tree-based state tying for high accuracy acoustic modeling. In *Proceedings ARPA Human Language Technology Conference*, Plainsboro, NJ, USA.
`ftp://svr-ftp.eng.cam.ac.uk/pub/reports/young_hlt94.ps.Z`

Young, S. (1995). Large vocabulary continuous speech recognition: a review. In *Proceedings IEEE Workshop on Speech Recognition*, Snowbird, Utah, USA.
`ftp://svr-ftp.eng.cam.ac.uk/pub/reports/young_LVRreview.ps.gz`

Young, S.J., and L.L. Chase (1998). Speech recognition evaluation: a review of the U.S. CSR and LVCSR programmes. *Computer Speech and Language*, **12**, 263–279.

Young, S., G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev and Ph. Woodland (2002). *The HTK Book (for HTK Version 3.2.1)*. CUED Speech Group, Cambridge, UK.
`http://htk.eng.cam.ac.uk/prot-docs/htk_book.shtml`

D. Šilingas is a PhD student in Department of Applied Informatics at Vytautas Magnus University since 2001. His research field is self-learning methods for Lithuanian speech recognition. The other research interests include software development methodologies and applications of Unified Modeling Language. Darius Šilingas is also a head of Training Department at joint-stock company "Baltijos programine iranga", where he takes responsibility of managing various training courses for IT professionals.

L. Telksnys is a professor, doctor habilitatis in informatics, a member of the Lithuanian Academy of Sciences. He is a head of Recognition Processes Department and a head of UNESCO Chair in Informatics for Humanities at the Institute of Mathematics and Informatics, professor at Vytautas Magnus University. Laimutis Telksnys is the author of an original theory of detecting changes in random processes, investigator and developer of computerized systems for statistical analysis and recognition of random signals. He is the author of 155 scientific publications and 6 inventions. His current research interests are speech processing, analysis and recognition of random processes, computerised multimedia systems.

Paslėptųjų Markovo modelių modifikacijų, naudojamų didelio žodyno tolydžios šnekos atpažinimui, savybės

Darius ŠILINGAS, Laimutis TELKSNYS

Straipsnyje nagrinėjamos šnekos atpažinimo, naudojant paslėptuosius Markovo modelius, savybės. Aptariama paprastų ir kontekstinių fonemų naudojimo, požymių pasiskirstymo modeliavimo Gauso bei dviejų ir trijų komponentų Gauso mišinių tikimybės pasiskirstymo funkcijomis, kalbos modelio integravimo įtaka. Modifikacijų tinkamumą praktiniams pritaikymams įvertinti naudojami žodžių atpažinimo tikslumo bei modelių sudėtingumo kriterijai. Aprašoma didelio žodyno tolydžios šnekos atpažinimo sistema, sukurta naudojant HTK porgraminę įrangą bei WSJCAM0 anglų šnekos duomenų bazę. Pristatomi eksperimentinių tyrimų rezultatai.