Available online at www.sciencedirect.com

SCIENCE DIRECT®

Speech Communication 48 (2006) 1214–1226

SPEECH COMMUNICATION

www.elsevier.com/locate/specom

A lattice search technique for a long-contextual-span hidden trajectory model of speech ☆

Dong Yu *, Li Deng, Alex Acero

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Received 21 November 2005; received in revised form 2 May 2006; accepted 10 May 2006

Abstract

We have recently developed a long-contextual-span hidden trajectory model (HTM) which captures underlying dynamic structure of speech coarticulation and reduction. Due to the long-span nature of the HTM and the complexity of its likelihood score computation, *N*-best list rescoring was the principal paradigm for evaluating the HTM for phonetic recognition in our earlier work. In this paper, we describe improved likelihood score computation in the HTM and a novel A^*-based time-asynchronous lattice-constrained decoding algorithm for the HTM evaluation. We focus on several special considerations in the decoder design, which are necessitated by the dependency of the HTM score at each given frame on the model parameters associated with a variable number of adjacent past and future phones. We present details on how the nodes and links in the lattices are expanded via a look-ahead mechanism, on how the A^* heuristics are estimated, and on how pruning strategies are applied to speed up the search process. The experiments on the standard TIMIT phonetic recognition task show improvement of recognition accuracy by the new search algorithm on recognition lattices over the traditional *N*-best rescoring paradigm.
© 2006 Elsevier B.V. All rights reserved.

Keywords: A^* search over recognition lattices; Decoder; Phonetic recognition; Vocal tract resonances; Speech dynamics; Hidden trajectories; Contextual assimilation; Filtering of targets; TIMIT; Long-span context dependence; Lattice rescoring; Pruning; Speech recognition

1. Introduction

Modeling long-span contextual effects in speech acoustics using underlying speech production mechanisms is a novel direction for speech recognition, especially for spontaneous speech where phonetic reduction and coarticulation are often mixed (e.g., Bakis, 1991; Deng and Braam, 1994; Rose et al., 1996; Bridle et al., 1998; Deng, 1998, 2004; Richards and Bridle, 1999; Gao et al., 2000; Sun and Deng, 2002; Deng and O'Shaughnessy, 2003; Ma and

☆ This is an extended and significantly enhanced version of the paper "Evaluation of a Long-Contextual-Span Hidden Trajectory Model and Phonetic Recognizer Using A^* Lattice Search" published in Proceedings of Interspeech 2005, Lisbon, Portugal (Yu et al., 2005).

* Corresponding author. Tel.: +1 425 707 9282; fax: +1 425 706 7329.

E-mail addresses: dongyu@microsoft.com (D. Yu), deng@microsoft.com (L. Deng), alexac@microsoft.com (A. Acero).

0167-6393/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2006.05.002

Deng, 2003; Zhou et al., 2003; Bilmes, 2004). Recently, we developed a long-contextual-span hidden trajectory model (HTM) based on bi-directional target filtering and a nonlinear transformation for cepstral that jointly model speech coarticulation and contextually assimilated reduction (Deng et al., 2004, 2005a,b; Yu et al., 2005). In this model, each phone-like unit in the HTM "phone" set is assumed to have a probabilistic target vector, which follows a Gaussian distribution. Hidden probabilistic vocal tract resonance (VTR) trajectories are computed by filtering the segmental VTR target sequence through a bi-directional non-causal finite-impulse-response filter. A nonlinear mapping with statistically characterized residuals generates probabilistic cepstral trajectories from the VTR trajectories. The statistical characterization of the HTM allows for straightforward computation of the model likelihood score for the cepstral observation data given the phone sequence and phone segment boundaries.

This bi-directional filter based HTM functionally achieves both anticipatory and regressive coarticulation, while keeping the phonological representation as the linear phonemic sequence and bypassing the use of more elaborate nonlinear phonological constructs. Different from other types of the acoustic models such as the hidden Markov model (HMM) that often uses a large set of context-dependent parameters with short-span modeling capabilities, the HTM uses a compact set of context-independent parameters to represent long-span contextual effects in the acoustic model through the technique of parametric filtering on VTR. In other words, the HTM represents long-span contextual dependencies without introducing additional model parameters by enumerating the context as is done for the HMM.

The long-contextual-span nature of the HTM, however, presents special challenges for decoding; that is, the search space increases dramatically and the nodes in the search space can span over a variable number of phones and frames. To avoid this difficulty, we evaluated the HTM performance using *N*-best rescoring in our earlier work. The work presented in this paper is aimed at confronting the above decoding difficulty directly. To this end, we have implemented an A^* based time-asynchronous lattice search algorithm for the HTM. We present in this paper details on (1) the expansion of the nodes and links in the lattices via look-ahead; (2) estimation of the A^* heuristics; and (3) pruning strategies to speed up the search. Our experiments on TIMIT phonetic recognition show that our new lattice search algorithm brings the core set phone recognition accuracy to 75.1% compared with the best earlier result of 74.6% using 1000-best rescoring.

The organization of this paper is as follows. In Section 2, we outline the bi-directional FIR-based target filtering framework, and describe the improved integrated likelihood score computation for the HTM. In Section 3, we present the new lattice-constrained search algorithm, heuristic score estimation, and the node expansion algorithm. In Section 4, speed-up techniques such as caching and pruning are discussed. We show the TIMIT phone recognition experimental results to demonstrate the effectiveness of the HTM and the new search algorithm in Section 5, and conclude the paper in Section 6.

2. The hidden trajectory model

Our hidden trajectory model (HTM) as a statistical generative model involves three constructive steps conceptually. In the first step, a phone-sequence hypothesis (including segment timing) is mapped into an HTM phone sequence (see Section 2.1) and its corresponding VTR targets based on the phone segment information. In the second step, the model predicts the VTR trajectory corresponding to the phone-sequence hypothesis by applying a bi-directional filter on the target sequence (see Section 2.2). In the third step, the model converts the estimated VTR trajectory into the cepstral trajectory with a non-linear mapping function (see Section 2.3). A likelihood score is then calculated by comparing the estimated cepstral trajectory with the (observed) cepstral trajectory derived directly from the waveform (see Section 2.4). In our original implementation, the system carried out these steps in a serial manner and the targets were treated as deterministic vectors. Recently, we improved the system by taking into consideration the VTR target variances, which give rise to probabilistic VTR trajectories. This leads to an integrated calculation of the acoustic likelihood score that we use for the lattice-based decoder design and implementation in the current work.

2.1. Converting phone sequence to target sequence

To predict the VTR trajectory of a phone sequence, we first convert the regular phone-like units into the "HTM phone set". Table 1 shows the conversion rules for the TIMIT phone-like units. There are four types of the phone-like units in Table 1. The first type keeps its original phone-like units' names. For example, phone *sh* is mapped to *sh* in the HTM phone set. The second type maps to different phones based on the phone that follows it. If the next phone is one of the front vowels (i.e., *ae*, *eh*, *ih*, *iy*, *y*, and *ey*), the phone is mapped to the "front" version of the phone. Otherwise, it is mapped to the phone with the same name (i.e., "non-front"). For example, unit *b* in TIMIT may be mapped to phone *b_f* or *b* based on the next phone. The third type of the units consists of composite phones including diphthongs and affricates. Each composite phone segment is split into two HTM phones around the middle of the segment. For example, phone *aw* becomes *aw*1 and *aw*2 in the HTM phone set. The fourth type of units is transformed to the HTM phone based on the specific rules defined in Table 1. For example, phone *gcl* becomes HTM phone *vcl*.

Once the phone-like units are transformed into HTM phones, we assume that each phone S (except for units *sil*, *cl*, *vcl*, *hh*, and *sp* which inherit targets from the phone following it) is associated with a multidimensional target vector $\boldsymbol{t}_s$ with a Gaussian distribution:

$$p(\boldsymbol{t}_s) = p(\boldsymbol{t}|s) = N(\boldsymbol{t}; \boldsymbol{\mu}_{T_s}, \boldsymbol{\Sigma}_{T_s}), \tag{1}$$

where $\boldsymbol{\mu}_{T_s}$ is the mean of the Gaussian distribution, and $\boldsymbol{\Sigma}_{T_s}$ is the covariance matrix of the Gaussian distribution. Note that in the current model implementation, the Gaussian in (1) around the target mean vector is sampled once per frame, not once per segment. Each target $\boldsymbol{t}$ consists of P resonant frequencies, $\boldsymbol{f}$, followed by P bandwidths, $\boldsymbol{b}$. That is,

$$\boldsymbol{t} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{b} \end{pmatrix}, \tag{2}$$

where

$$\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_P \end{pmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_P \end{pmatrix}. \tag{3}$$

Thus, given the phone sequence and its boundaries, we can form a sequence of targets which is a function of discrete time and which jumps at the segments' boundaries. Mathematically, the target sequence can be represented as a sequence of step-wise constant functions with variable durations and heights:

Table 1
Map original phone set to HTM phone set

Type	Description	Original phone	Conversion rule
1	Phones unchanged after the mapping	*d*, *t*, *dx*, *vcl*, *cl*, *s*, *sh*, *z*, *zh*, *th*, *dh*, *n*, *l*, *r*, *w*, *y*, *hh*, *iy*, *ih*, *eh*, *ae*, *aa*, *ah*, *uh*, *uw*, *er*, *ax*, *sil*, *sp*	$ph \to ph$
2	Phones mapped to different HTM phones based on the phone that follows it (subscript *_f* denotes 'front-vowel context')	*b*, *g*, *p*, *f*, *k*, *m*, *ng*, *v*	if next_$ph \in \{ae, eh, ih, iy, y, ey1\}$ $ph \to ph\_f$ otherwise $ph \to ph$
3	Composite phones	*jh, ch, ey, aw, ay, oy, ow,*	$ph \to ph1 + ph2$
4	Phones follow specific conversion rules	*q*, *kcl*, *pcl*, *tcl*	$ph \to cl$
		bcl, *dcl*, *gcl*	$ph \to vcl$
		em	$ph \to m$
		En, *nx*	$ph \to n$
		eng	$ph \to eng$
		hv	$ph \to hh$
		el	$ph \to l$
		ao	$ph \to aa$
		ux	$ph \to uw$
		ix, *ax-h*	$ph \to ax$
		axr	$ph \to er$
		pau, *h#*, $\langle s \rangle$, $\langle /s \rangle$	$ph \to sil$
		epi	$ph \to sp$

$$\boldsymbol{t}(k) = \sum_{i=1}^{I} \left[u(k - k^{\mathrm{l}}_{s_i}) - u(k - k^{\mathrm{r}}_{s_i})\right] \boldsymbol{t}_{s_i}, \tag{4}$$

where $u(k)$ is the unit step function, k^{r}_s, $s = s_1, s_2, \ldots, s_I$ are the right boundary sequence of the P segments in the utterance, and k^{l}_s, $s = s_1, s_2, \ldots, s_I$ are the left boundary sequence. In general, $k^{\mathrm{l}}_{s_{i+1}} = k^{\mathrm{r}}_{s_i}$ for $1 \leqslant i < I$. The difference of the two gives the duration sequence. $\boldsymbol{t}_s$, $s = s_1, s_2, \ldots, s_I$ are the target vectors for the corresponding HTM "phone" unit segments.

2.2. Converting target sequence to VTR trajectory

The target sequence is taken as the input of a slowly time-varying, bi-directional FIR filter to estimate the VTR trajectories. This filter is characterized by the following non-causal, vector-valued, impulse response function, which is temporally symmetric:

$$h_s(k) = \begin{cases} c\gamma_{s(k)}^{-k}, & -D < k < 0, \\ c, & k = 0, \\ c\gamma_{s(k)}^{k}, & 0 < k < D, \end{cases} \tag{5}$$

where k represents time frame, each of which is typically 10 ms. An impulse response is the output of the filter in response to a delta function, and it characterizes full properties of the linear filter (Oppenheim et al., 1999). For the specific filter's impulse response in (5), $\gamma_{s(k)}$ is the *stiffness* parameter vector, one component for each VTR order (which spans F1–F4 in our case). Each component of the stiffness vector is a positive real value between zero and one. The subscript $s(k)$ in $\gamma_{s(k)}$ indicates that the stiffness vector is dependent on the segment state $s(k)$, which varies over time. D in (5) is the unidirectional length of the impulse response, representing a limit on the temporal extent of coarticulation in the temporal direction. D is assumed for simplicity to be equal in length for the forward direction (anticipatory coarticulation) and the backward direction (regressive coarticulation). In principle, D can be extended to the beginning and end of each utterance. But for the feasibility of model implementation, D is limited to be within the utterance.

In (5), c is the normalization constant to ensure that the filter weights add up to one. For simplicity, we make the assumption that over the temporal span $-D \leqslant k \leqslant D$, the stiffness parameter's value stays approximately constant

$$\gamma_{s(k)} \approx \gamma_{s(0)}. \tag{6}$$

That is, the adjacent segments within the temporal span $2D + 1$ in length which contribute to the coarticulated home segment have the same stiffness parameter value as that of the home segment. Under this assumption, we can determine c by requiring that the filter coefficients sum to one:

$$\begin{aligned} \sum_{k=-D}^{D} h_s(k) &= c \sum_{k=-D}^{D} \gamma_{s(k)}^{|k|} \\ &\approx c\left[1 + 2\left(\gamma_{s(0)} + \gamma_{s(0)}^2 + \cdots + \gamma_{s(0)}^D\right)\right] \\ &= c\,\frac{1 + \gamma_{s(0)} - 2\gamma_{s(0)}^{D+1}}{1 - \gamma_{s(0)}} = 1. \end{aligned} \tag{7}$$

Thus,

$$c_\gamma = c_{\gamma_{s(0)}} \approx \frac{1 - \gamma_{s(0)}}{1 + \gamma_{s(0)} - 2\gamma_{s(0)}^{D+1}}. \tag{8}$$

Given the filter's impulse response and the input to the filter, the filter's output as the model's prediction for the VTR trajectories is the convolution between these two signals. The result of the convolution within the boundaries of the home segment s is

$$\boldsymbol{z}_s(k) = \boldsymbol{h}_{s(k)} \otimes \boldsymbol{t}(k) = \sum_{\tau=k-D}^{k+D} c(\gamma_{s(\tau)}) \boldsymbol{t}_{s(\tau)} \gamma_{s(\tau)}^{|k-\tau|} = \boldsymbol{a}_k \cdot \boldsymbol{t}_{s(\tau)}, \tag{9}$$

where k is the index of an output frame, τ is the index of an input frame, and the target vector and the input target vector $\boldsymbol{t}(k)$ and the filter's stiffness vector $\gamma_{s(\tau)}$ may take not only values associated with the current home segment, but also those associated with the adjacent segments since the time τ in (9) can go beyond the home segment's boundaries. The last part of (9) is a vector notation that compactly represents the coarticulation effects of target filtering by a time-dependent vector $\boldsymbol{a}_k$. This notation facilitates the derivation of the model training algorithm as outlined in (Deng et al., 2005b), which will not be discussed in this paper.

A sequential concatenation of all outputs $\boldsymbol{z}_s(k)$, $s = s_1, s_2, \ldots, s_P$ constitutes the model prediction of VTR trajectories for the entire utterance:

$$\boldsymbol{z}(k) = \sum_{i=1}^{P} \left[u(k - k^{\mathrm{l}}_{s_i}) - u(k - k^{\mathrm{r}}_{s_i})\right] \boldsymbol{z}_{s_i}(k), \tag{10}$$

where $u(k)$ is a unit step function. An example of the course of the VTR targets (for three distinct phones) and their filtered trajectories will be shown later in Section 3.2 (Fig. 4).

Note that the convolution operation above carried out by the filter in the model smoothes out the trajectories at each junction of two adjacent segments, contrasting the discontinuous jump in the input to the filter at the same junction. The smoothness is applied to all classes of speech sounds including consonantal closure. This provides the mechanism for coarticulation and VTR target undershooting in the current hidden trajectory model.

The bi-directional filter gives rise to both forward and backward coarticulation, since it makes the VTR value at each time dependent not only on the current phone's VTR target but also on the VTR targets of the adjacent phones. This filtering process has been shown to give quantitative prediction of the magnitude of contextually assimilated reduction and coarticulation (Deng et al., 2004).

Since the VTR vector $\boldsymbol{z}(k)$ (at each frame k) is a linear function of the targets $\boldsymbol{t}$ that follows a Gaussian distribution, $\boldsymbol{z}(k)$ is a Gaussian as well:

$$p(\boldsymbol{z}(k)|s) = N\left[\boldsymbol{z}(k); \boldsymbol{\mu}_{z(k)}, \boldsymbol{\Sigma}_{z(k)}\right], \tag{11}$$

whose mean vector $\boldsymbol{\mu}_{z(k)}$ is

$$\boldsymbol{\mu}_{z(k)} = \sum_{\tau=k-D}^{k+D} c_\gamma \gamma_{s(\tau)}^{|k-\tau|} \boldsymbol{\mu}_{T_{s(\tau)}} = \boldsymbol{a}_k \cdot \boldsymbol{\mu}_{T_{s(\tau)}}, \tag{12}$$

and the covariance matrix is

$$\boldsymbol{\Sigma}_{z(k)} = \sum_{\tau=k-D}^{k+D} c_\gamma^2 \gamma_{s(\tau)}^{2|k-\tau|} \boldsymbol{\Sigma}_{T_{s(\tau)}} = \boldsymbol{v}_k \cdot \boldsymbol{\Sigma}_{T_{s(\tau)}}. \tag{13}$$

In (12), $\boldsymbol{a}_k$ is the time-dependent "coarticulation vector" as in (9). And in (13), $\boldsymbol{v}_k$ is the vector which is the element-wise square of vector $\boldsymbol{a}_k$. In our current implementation, both the target covariance matrix $\boldsymbol{\Sigma}_{T_s}$ and the VTR covariance matrix $\boldsymbol{\Sigma}_{z(k)}$ for each phone segment are approximated by diagonal matrices. In (13), the dot product applies to vector $\boldsymbol{v}_k$ and the diagonal elements of $\boldsymbol{\Sigma}_{T_s}$ treated as a vector.

2.3. Converting VTR trajectory to cepstral trajectory

The mapping between the VTR vector $\boldsymbol{z}(k)$ (which consists of both VTR frequency sub-vector $\boldsymbol{f}(k)$ and bandwidth sub-vector $\boldsymbol{b}(k)$) and the estimated corresponding vector of LPC cepstral $\hat{\boldsymbol{o}}(k)$ can be represented by

$$\begin{aligned}\hat{o}_j(k) &= (F[\boldsymbol{z}_k])_j = F_j(k) \\ &= \frac{2}{j}\sum_{p=1}^{P} \exp\left(-\pi j \frac{b_p(k)}{f_s}\right) \cos\left(2\pi j \frac{f_p(k)}{f_s}\right),\end{aligned} \tag{14}$$

where P is the highest VTR order ($P = 4$ in the current implementation) and f_s is the sampling rate (e.g., $f_s = 16{,}000$ in the TIMIT dataset). $f_p(k)$ and $b_p(k)$ are the frequency and bandwidth of the pth resonance at frame k. See a detailed derivation of (14) in (Deng et al., 2006a) using the all-pole assumption for the speech signal.

We model the cepstral prediction's *residual* vector, defined by the difference between the observed acoustic (cepstral) vector $\boldsymbol{o}(k)$ and its predicted value:

$$\boldsymbol{r}_s(k) = \boldsymbol{o}(k) - F[\boldsymbol{z}(k)], \tag{15}$$

as a Gaussian distribution

$$p(\boldsymbol{r}_s(k)|\boldsymbol{z}(k), s) = N\left[\boldsymbol{r}_s(k); \boldsymbol{\mu}_{r_s(k)}, \boldsymbol{\Sigma}_{r_s(k)}\right], \tag{16}$$

whose mean vector is denoted by $\boldsymbol{\mu}_{r_s(k)}$ and the covariance matrix by $\boldsymbol{\Sigma}_{r_s(k)}$. The conditional distribution of the cepstral vector is thus:

$$p(\boldsymbol{o}(k)|\boldsymbol{z}(k), s) = N\left[\boldsymbol{o}(k); F[\boldsymbol{z}(k)] + \boldsymbol{\mu}_{r_s(k)}, \boldsymbol{\Sigma}_{r_s(k)}\right]. \tag{17}$$

In order to compute the acoustic observation likelihood, we linearize the nonlinear mean function of $F[\boldsymbol{z}(k)]$ by using the first-order Taylor series approximation:

$$F[\boldsymbol{z}(k)] \approx F[\boldsymbol{z}_0(k)] + F'[\boldsymbol{z}_0(k)](\boldsymbol{z}(k) - \boldsymbol{z}_0(k)), \tag{18}$$

where $\boldsymbol{z}_0(k)$ is the Taylor series expansion point (obtained by a fast VTR tracker), and the components of Jacobian matrix $F'[\cdot]$ is computed (using standard calculus) in a closed form as

$$F'_j[f_p(k)] = -\frac{4\pi}{f_s} \exp\left(-\pi j \frac{b_p(k)}{f_s}\right) \sin\left(2\pi j \frac{f_p(k)}{f_s}\right), \tag{19}$$

for the VTR frequency components of $\boldsymbol{z}$, and

$$F'_j[b_p(k)] = -\frac{2\pi}{f_s}\exp\left(-\pi j\frac{b_p(k)}{f_s}\right)\cos\left(2\pi j\frac{f_p(k)}{f_s}\right), \tag{20}$$

for the bandwidth components of $\boldsymbol{z}$.

We can thus obtain the approximation:

$$p(\boldsymbol{o}(k)|\boldsymbol{z}(k),s) \approx N\left[\boldsymbol{o}(k);\boldsymbol{\mu}_{o_s(k)},\boldsymbol{\Sigma}_{r_s(k)}\right], \tag{21}$$

where

$$\boldsymbol{\mu}_{o_s(k)} = F'[\boldsymbol{z}_0(k)]\boldsymbol{z}(k) + (F[\boldsymbol{z}_0(k)] - F'[\boldsymbol{z}_0(k)]\boldsymbol{z}_0(k) + \boldsymbol{\mu}_{r_s(k)}) \tag{22}$$

2.4. Computing likelihood score

The major goal of HTM model construction is to determine the likelihood value of the acoustic observation vector $\boldsymbol{o}(k)$ given an arbitrary phone sequence hypothesis. Once the model provides the likelihood value for each possible phone sequence (with timing), then recognition is a matter of searching for the optimal phone sequence (the one with the highest likelihood), i.e.,

$$\hat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} p(\boldsymbol{o}|\boldsymbol{w}). \tag{23}$$

When computing the HTM likelihood score, the VTR vector $\boldsymbol{z}(k)$ is treated as hidden variables which are marginalized (i.e., integrated over) in the likelihood computation. The final result of the computation is as follows:

$$\begin{aligned} p(\boldsymbol{o}(k)|s) &= \int p[\boldsymbol{o}(k)|\boldsymbol{z}(k),s]p[\boldsymbol{z}(k)|s]\,\mathrm{d}\boldsymbol{z} \\ &= N\left[\boldsymbol{o}(k);\bar{\boldsymbol{\mu}}_{o_s(k)},\overline{\boldsymbol{\Sigma}}_{o_s(k)}\right], \end{aligned} \tag{24}$$

where the time-varying mean vector is

$$\bar{\boldsymbol{\mu}}_{o_s(k)} = F[\boldsymbol{z}_0(k)] + F'[\boldsymbol{z}_0(k)][\boldsymbol{a}_k\cdot\boldsymbol{\mu}_T - \boldsymbol{z}_0(k)] + \boldsymbol{\mu}_{r_s(k)}, \tag{25}$$

and the time-varying covariance matrix is

$$\overline{\boldsymbol{\Sigma}}_{o_s(k)} = \overline{\boldsymbol{\Sigma}}_{r_s(k)} + F'[\boldsymbol{z}_0(k)]\boldsymbol{\Sigma}_{z(k)}(F'[\boldsymbol{z}_0(k)])^{\mathrm{Tr}}. \tag{26}$$

Note that (24) computes the likelihood of one observation frame, as indicated by frame index k. To compute the likelihood score for the entire utterance, the frame-level likelihoods are multiplied.

In our current implementation, the stiffness factor, γ, and the number of context frames, D, are pre-chosen. They are fixed for all phones and resonances with the same value. All other parameters are trained automatically. Interested readers may refer to (Deng et al., 2005b) for details of the parameter learning algorithm.

3. Lattice search algorithm

In addition to the general difficulties of decoding for speech recognition with short-contextual-span HMMs (Aubert, 2002; Ortmanns and Ney, 2000), the long-contextual-span nature of the HTM presents three special difficulties for decoder design. First, the HTM likelihood of each observation frame depends on the contextual phone hypothesis associated with left D (or more) frames and right D (or more) frames from the current one. In many cases, more than D frames are actually invoked since the “substituted target” values of the targetless phones such as *closure*, *pause*, *silence*, and phone */hh/* are taken as the VTR targets of the phones immediately following them. (In addition, the VTR targets for the labial, labial–dental, and velar consonants are made dependent on whether the following phone is a front vowel or not; see Table 1.) A naïve calculation indicates that each frame would be expanded to as many as 60^{15} context dependent frames for a 60 phone system with $D = 7$. Second, each frame may depend on a variable length of frames and variable number of phones for different contexts. Third, the segment likelihood score can be directly calculated only when all the hypothesized phone boundaries are available.

Due to these difficulties, we had been able to evaluate the HTM with only N-best rescoring prior to this work. N-best rescoring has less of the above problems. The N-best approach, unfortunately, has the disadvantage of severely limited hypotheses for rescoring. In this section, we describe our A* based lattice search and rescoring algorithm that dramatically expands the limited number of hypotheses that would be imposed by the N-best rescoring paradigm. Lattice rescoring has the potential to evaluate a very large number of hypotheses and to achieve better accuracy than N-best rescoring. For standard HMM-based speech recognition, the lattice-constrained search has also been successfully used to reduce the full complexity of the unconstrained search (e.g., Goel and Byrne, 1999).

3.1. Phonetic lattice

A lattice is an acyclic directed graph $(\mathcal{N}, \mathcal{A})$ with one start node $!s$ and one end node $!e$, where $\mathcal{N}$ is the set of nodes, and $\mathcal{A}$ is the set of directed arcs. Each arc a in set $\mathcal{A}$ has a start node $a.s$ and an end node $a.e$. A lattice can be considered as a compact representation of a very large N-best list. Since our long-contextual-span HTM is phone based, the rescoring is most naturally conducted on the phone lattice. (This can be extended to the word lattice easily but will not be addressed in this paper.) In the implementation of phone lattice search, we adopt the HTK lattice format. In this format, each node of the lattice consists of two elements: (w, t), where w is the identity of a phone, and t is the time stamp for the end time of phone w (which is the same as the start time of the next phone).

A partial phone-sequence hypothesis constrained by a lattice is a sequence of connected nodes (phones) starting from start-node $!s$. A phone-sequence hypothesis is complete if the sequence also ends at end-node $!e$. We call a partial hypothesis h_1 a prefix of another partial hypothesis h_2 if h_1 is the head of h_2, i.e., $h_2 = h_1 \circ x$ for some node sequence x. Note that two sequences with the same phone identities but with different phone boundaries are considered as two different hypotheses, and they follow two different paths in the lattice. In other words, a lattice represents a sparse subset of all possible phone sequences with all possible phone ending times. Fig. 1 depicts an example of a very small phonetic lattice. Notice in the Fig. 1 that $(ae, 8)$ and $(ae, 7)$ are two different nodes in the lattice, and $(!s, 0) \circ (p, 3) \circ (ae, 8)$ and $(!s, 0) \circ (p, 3) \circ (ae, 7)$ are two different partial hypotheses. There are only three complete hypotheses in the lattice, i.e.,

$$(!s, 0) \circ (p, 3) \circ (ae, 8) \circ (iy, 16) \circ (d, 21) \circ (!e, 27),$$
$$(!s, 0) \circ (p, 3) \circ (ae, 7) \circ (iy, 17) \circ (!e, 27), \quad \text{and}$$
$$(!s, 0) \circ (b, 4) \circ (ae, 10) \circ (d, 19) \circ (!e, 27).$$

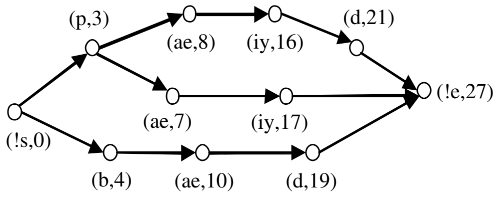

Fig. 1. An example of a phonetic lattice.

In our current implementation, we rescore the lattices generated by a traditional HMM tri-phone model (although the same algorithm can be extended to other lattices). As we have mentioned earlier, in the long-span HTM, the score of each phone (arc) depends on the hypothesized VTR targets of at least past and future D frames. This may include several phones in either temporal direction and the number of phones in the context is variable. In other words, we cannot simply use the nodes and arcs in the original lattice for the likelihood computation and book keeping. Instead, we need to convert each node into a set of variable-spanning context-dependent nodes (VSCD-node). Each VSCD-node is a triplet: $\{n_p, n_c, n_f\}$, where n_p is a list of past nodes in the original lattice, n_c is the center node under consideration, and n_f is a list of future nodes in the original lattice. The number of nodes expanded depends on the parameter D. As an illustrating example, if $D = 5$, node $(p, 3)$ in Fig. 1 would be expanded into two VSCD-nodes:

$$\{(!s, 0), (p, 3), (ae, 8)\} \quad \text{and}$$
$$\{(!s, 0), (p, 3), (ae, 7) \circ (iy, 17)\},$$

and the single arc from $(!s, 0)$ to $(p, 3)$ in Fig. 1 becomes two arcs as shown in Fig. 2.

In this expanded lattice, VSCD-node $\{n_{1,p}, n_{1,c}, n_{1,f}\}$ has an arc to VSCD-node $\{n_{2,p}, n_{2,c}, n_{2,f}\}$ if and only if there is an arc from $n_{1,c}$ to $n_{2,c}$ in the original lattice, and $n_{1,f}$ is a prefix of $n_{2,c} \circ n_{2,f}$ (which implies that $n_{2,c}$ is the first node in $n_{1,f}$). A partial hypothesis in our HTM thus can also be represented as a 2-tuplet: $[hyp, n_f]$, where hyp is the partial hypothesis without context and n_f is a list of future nodes, all represented as nodes in the original lattice. In other words, the HTM needs to look ahead to all phones in the n_f to be able to calculate the likelihood score for the partial hypothesis hyp.

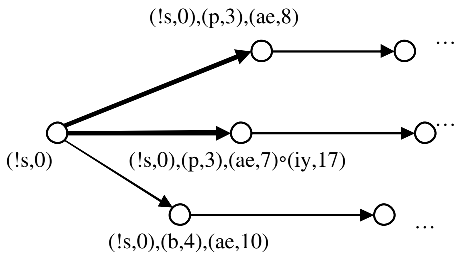

Fig. 2. Partially expanded phonetic lattice. Each node in this expanded lattice is a VSCD-node. Arcs are also multiplicatively expanded because the expansion needs to take into account both the past and future phones.

3.2. A* search

The goal of the lattice search is to find a complete hypothesis whose log-likelihood score is the highest among all the complete hypotheses constrained by the lattice (Goel and Byrne, 1999). In our implementation, the search over the expanded lattice is conducted using the one-stack A* search algorithm described in Fig. 3.

It has been well known (e.g., Russell and Norvig, 1995) that an A* search algorithm will always complete the best-scoring hypothesis first if the following two properties are satisfied:

Property 1. The estimated total score of a complete hypothesis is the true score.

Property 2. The estimated total score of a partial hypothesis is an upper bound of the true score of all the hypotheses whose prefix is the partial hypothesis.

If the estimated total score of a partial hypothesis is much larger than the best possible true score of all the hypotheses whose prefix is the partial hypothesis, then the estimation of the backward heuristic score would not provide useful information and the speed performance of an A* search would be low and be close to that of a breadth first search. On the other hand, if the estimated total score equals the best possible true score, the A* search will find the best complete hypothesis rapidly by expanding only the arcs along the best complete hypothesis.

```
Estimate heuristic scores for each arc;
Estimate the backward heuristic scores for each node;
Maintain a stack of partial hypotheses ordered on the
  estimated total log-likelihood in descending order;
Initialize the stack with !s - the start node of the lattice;
While the stack is not empty
   Pop up the top partial hypothesis from the stack;
   If the hypothesis is complete
      Return the hypothesis;
   Else
      Expand the hypothesis by one more phone;
      Calculate the HTM scores for the newly
        expanded phones;
      Calculate the estimated total score of the newly
        expanded partial hypotheses;
      Insert these new partial hypotheses to the stack at the
        appropriate positions;
   End If
End While
```

Fig. 3. The basic one-stack A* search algorithm.

3.3. Score estimation

As indicated in Section 3.2, the performance of the A* search algorithm depends heavily on the quality of the estimated total score. In the long-contextual-span HTM, the true score of a complete hypothesis is the sum of log-likelihoods over all the phones in the hypothesis as indicated in (27):

$$S = \sum_{i=1}^{N} S_i, \tag{27}$$

where S_i is the log-likelihood of the ith phone in the hypothesis, and N is the total number of phones in the hypothesis. When we only have a partial hypothesis, the total score S can be estimated as

$$\widehat{S} = S_p + H_n. \tag{28}$$

where S_p is the log-likelihood score of the partial hypothesis, and H_n is the backward heuristic score associated with the final node n (phone) in the partial hypothesis. It is obvious that the quality of the score estimation using (28) depends on the quality of the heuristic score H_n.

To ensure that the backward heuristic score is a good estimate of the true backward score, we developed a novel score computation algorithm which we describe next. In the long-span HTM based on bi-directional target filtering, the true score of each phone S_a, associated with an arc in the lattice, is a function of both the past and future VTR targets. This makes it very expensive to use the true future score as the heuristic score in (28). When we started this work, we first used the adjusted HMM score as the heuristic score since it is very efficient to obtain (already available in the lattice). However, the HMM score is very different from the HTM score and thus is a bad heuristic score. Our current method is based on estimating the true score by using the same formula for the true HTM score but without considering the adjacent phone context. Note that the heuristic score obtained in this way may be lower than the true score. This is especially true for the correct phone hypotheses in a phone context since the omission of the phone context information is likely to increase the mismatch between the acoustic signal and the hypothesis. This situation is shown in Fig. 4, where a generic context is used to replace the realistic context. The generic context takes the form of neutral VTR targets, which is independent of the adjacent phones. To compensate for the reduction of the score caused

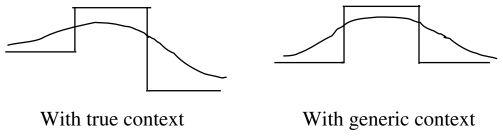

Fig. 4. An illustrative VTR trajectory with and without true context.

```
Sort the nodes in the descending order of time
For each node $n$
  If $n$ is the end node $!e$
    $H_n = 0$
  Else
    $H_n = -\infty$
    For each arc $a$ following node $n$
      $H_n = \max(H_n, H_a + H_{a.endN})$
    End For
  End If
End For
```

Fig. 5. Computing backward heuristic scores of nodes H_n.

by this approximation, we adjust the heuristic score by adding a fixed, small score for each frame so that the heuristic score will be highly likely to satisfy Property 2. This score is estimated from the training data. And we found in the experiments that the recognition results are relatively insensitive to its choice over a range of its values.

Once we have the heuristic scores of all the phones (arcs) H_a in the lattice, we can calculate the backward heuristic scores of each node H_n. To satisfy Property 2, H_n should be the best score along all possible paths starting from node n to the end. In Fig. 5, we describe the algorithm for calculating the backward heuristic scores H_n. The calculation of H_a and H_n incurs a very small computational cost, so is the sorting since much of the ordering information is already contained in the lattice.

3.4. Hypothesis expansion

One of the most important steps in search including A* search is expansion of the hypotheses. This expansion needs to be done in the expanded lattice (at the VSCD-node level). However, given the fact that a partial hypothesis in the HTM can be represented as a 2-tuplet: $[hyp, n_f]$, where hyp is the partial hypothesis without context and n_f is a list of future nodes, all represented as nodes in the original lattice, we can expand a partial hypothesis by directly traversing the original lattice according to the algorithm described in Fig. 6. It should be obvious that the partial score of a new hypothesis $[hyp \circ a.e, n_h]$ can be derived from the old hypothesis $[hyp, n_f]$ using

$$S_{p,new} = S_{p,hyp} + S_a. \tag{29}$$

```
For each out-going arc $a$ of the $hyp$'s last node $n_L$
  If $a.e$ is the first node in $n_f$
    Put $n_f$ - $a.e$ into the stack
    While the stack is not empty
      Pop up the list $n_h$ from the stack
      Set $h$ = the last node of $n_h$
      If $h$ is not $D$ frames ahead of $n_L$
        Append each out-going arcs of $h$ to $n_h$
        Put all new $n_h'$ into the stack
      Else
        $[hyp \circ a.e, n_h]$ is a new hypothesis
      End If
    End While
  End If
End For
```

Fig. 6. Hypothesis expansion. A partial hypothesis is a 2-tuplet: $[hyp, n_f]$, where hyp is the partial hypothesis without context and n_f is a list of future nodes, all represented as nodes in the original lattice. $\circ$ denotes concatenation.

4. Speed-up strategies for the search algorithm

The lattice rescoring can be very slow (depending on the size of the lattice) and it consumes much memory given a large number of possible expansions in the HTM. We used several strategies to speed up the rescoring process.

4.1. Score caching

Many times, the partial hypotheses are different but the VSCD-nodes contained in these hypotheses may have the same phone list and boundaries, a condition which is often called "collisions". Since the log-likelihood of a phone in our model solely depends on the phone list and corresponding boundaries in the VSCD-node $\{n_p, n_c, n_f\}$, we can cache the phone scores in a hash table to avoid re-computation. The hash value used in our system is derived from the VSCD-node's fingerprint, which consists of concatenated pairs of *phoneID:frameNumber* for each node in the VSCD-node with an additional number indicating the location of n_c in the VSCD-node. Score caching can eliminate about 80% of the calculation.

4.2. Heuristic score beam pruning

Beam pruning is one of the most important pruning strategies used in large vocabulary continuous speech recognition (LVCSR) decoders. The basic idea of beam pruning is to retain only paths with a likelihood score close to the best partial path hypothesis. However, this technology cannot be directly applied to time-asynchronous decoders since partial hypotheses in the time-asynchronous decoders contain different number of frames. In our decoder, we used a different flavor of beam pruning.

We have observed that the heuristic score of the true hypothesis is generally not very far away from the best heuristic score in the lattice. This is because the heuristic scores used in the decoder are still based on the HTM scores, except without specific context information. For this reason, our search algorithm prunes out those hypotheses whose heuristics are much lower than the best hypothesis in the lattice, eliminating the expansion and evaluation of these hypotheses. This pruning step is done before A* search starts. More formally, we define the best forward heuristic score of a node n as

$$H_n^{\mathrm{f}} = \max_{a.\mathrm{end}N=n}(H_a + H_{a.\mathrm{start}N}^{\mathrm{f}}), \tag{30}$$

where H_a is the heuristic score of arc a, and $H_{a.\mathrm{start}N}^{\mathrm{f}}$ is the forward heuristic score of node a.start$N$. The search algorithm prunes out all nodes and corresponding arcs that:

$$H_a + H_{a.\mathrm{start}N}^{\mathrm{f}} < f_{\mathrm{beam}} \cdot H_n^{\mathrm{f}}, \tag{31}$$

where f_{beam} is the threshold that determines the beam width.

Similarly, we define the best backward heuristic score of a node n as

$$H_n^{\mathrm{b}} = \max_{a.\mathrm{start}N=n}(H_a + H_{a.\mathrm{end}N}^{\mathrm{b}}). \tag{32}$$

and prune the nodes and arcs backwards.

4.3. Histogram pruning

Histogram pruning is another common pruning strategy. The basic idea of histogram pruning is to limit the number of active partial hypotheses at each step by retaining only a predefined number of best paths. In other words, the search algorithm sorts the partial hypotheses in the stack in the descending order of the estimated total likelihood scores $\widehat{S}(h)$. The partial hypothesis h is pruned out if:

$$Pos(h) > N_{\mathrm{hist}} \cap \widehat{S}(h) \neq \widehat{S}(h_i), \quad \forall i \leqslant N_{\mathrm{hist}}, \tag{33}$$

where $Pos(h)$ is the position of the hypothesis h in the stack, and N_{hist} is histogram pruning threshold.

Histogram pruning is more likely to prune out the best path than the beam pruning. For this reason, histogram pruning is mainly used to limit the stack size in our A* search algorithm. The histogram pruning threshold is determined based on the worst case processing time for decoding and the memory available.

4.4. Merge pruning

When several partial hypotheses join at the same VSCD-node, we only need to keep the hypothesis with the highest partial score (or the highest estimated total score since the heuristic part would be the same). This pruning strategy is based on the fact that complete hypotheses expanded from other partial hypotheses will not have a higher score than the complete hypothesis expanded from the partial hypothesis with the highest partial score. More formally, the search algorithm prunes out all hypotheses h_i that satisfy:

$$\widehat{S}(h_i) < \max_{h_{i.\mathrm{end}N}=h_{j.\mathrm{end}N}}(\widehat{S}(h_j)), \tag{34}$$

where $\widehat{S}(h_i)$ and $h_{i.\mathrm{end}N}$ are the estimated total score and the end VSCD-node of the hypotheses h_i respectively. Note, however, if the decoder needs to report N-best results, merge pruning cannot be applied.

4.5. Prefix pruning

The output of the decoder is a phone sequence without boundary information. In other words, we can ignore the boundary information during the pruning process and consider hypotheses with the same phone sequence but different boundaries the same. In our search process, we keep only two partial hypotheses: (1) with the highest partial score, and (2) with the highest estimated total score in the stack, among all having the same phone sequence and ending frame number but different boundaries. (These hypotheses do not need to merge at the same VSCD-node.) This pruning strategy is called prefix pruning since the hypotheses pruned out have the same prefix phone list.

Unlike the merging pruning, prefix pruning may lose accuracy since different hypotheses with the same prefix may link to different sets of future nodes which usually provide different scores for the future frames. The reason to prune only those hypotheses with the same ending frame number is to make sure the scores are comparable. The reason to keep both the hypotheses with the highest partial score and the highest estimated total score is to reduce the probability that the best path is pruned out by the prefix pruning.

4.6. Unseen phone bi-gram pruning

The basic idea behind the unseen phone bi-gram pruning is that the correct hypothesis is very unlikely to contain a large percentage of unseen phone bi-grams. This pruning strategy is applied in the hypothesis expansion step. The search algorithm will not put a newly generated partial hypothesis to the A* stack if the percentage of unseen bi-grams in the hypothesis exceeds a specific threshold. In other words, the search algorithm prunes out the partial hypothesis h if

$$\frac{B_u(h)}{B_t(h)} > \theta_u \cap B_t(h) > U_{\min}, \tag{35}$$

where $B_u(h)$ is the number of unseen bi-grams in the hypothesis h, $B_t(h)$ is the total number of bi-grams in the hypothesis, θ_u is the pruning threshold (in percentage, e.g., 10%), and $U_{\min}$ is the minimum number of bi-grams in the hypothesis before the unseen phone bi-gram pruning strategy is applied.

The bi-gram dictionary is obtained from the training set and stored in the prefix format. The larger the threshold, the less effect the pruning has, and the higher chance that the best hypothesis will not be pruned out. The pruning threshold θ_u is tuned using a held-out set.

5. Experimental results

We have conducted phonetic recognition experiments on the TIMIT standard core test set (192 utterances) as described in (Glass, 2003) to evaluate our long-contextual-span HTM. The phone lattices used in our experiments are generated by a conventional, high-quality tri-phone HMM with phone bigram as the "language model" (LM). The same LM is used for the HTM.

The complete log likelihood score used in the decoding process is the weighted score of (1) log HTM likelihood, (2) log HMM likelihood, (3) LM score, and (4) insertion penalty as follows:

$$S(h) = \lambda_{\text{htm}} \cdot S_{\text{htm}}(h) + \lambda_{\text{hmm}} \cdot S_{\text{hmm}}(h) + \lambda_{\text{lm}} \cdot S_{\text{lm}}(h) + \text{IP} \cdot N_{\text{phone}}(h), \tag{36}$$

where S_{htm},S_{hmm}, and S_{lm} are HTM score, HMM score, and LM score respectively, and λ_{htm}, λ_{hmm}, and λ_{lm} are their respective weights. IP is the insertion penalty and N_{phone} is the total number of phone-like units in the hypothesis being evaluated.

As the baseline, we first conducted the 1000-best list rescoring experiments using the HTM after LPCCs are warped in a Mel scale. The warping factor for testing is fixed at 0.48; see details of LPCC warping in (Oppenheim and Johnson, 1972), where the warping is mathematically represented by a matrix multiplication. The LPCC feature warping uses the same matrix multiplication and in the HTM, Eqs. (14)–(17) are modified accordingly. The phonetic recognition accuracy results for N-best rescoring scheme using this HTM on the Mel-warped LPCC features are listed in Table 2, where the sensitivity to the HMM weight and phone bi-gram LM is shown. The best phone accuracy obtained from using 1000-best list rescoring is 74.6% (i.e., phone error rate (PER) is 25.4%). The optimal value of IP = −0.5 is used.

The results of the lattice search algorithm described in Sections 3 and 4 on phonetic recognition are presented in Table 3. The initial lattice for each test utterance is generated from an HTK–HMM system, with the averaged total number of nodes in the order of 1000, the total number of links in the order of 8000, and the total number of expanded hypotheses in the order of 10^{30}. Despite its large size, this initial lattice is found to still contain 2% of oracle errors if boundary error is ignored. The accuracy of the top candidate in this HMM system is 72.5%. After various stages of pruning, the averaged total number of hypotheses evaluated is estimated to be

Table 2
N-best rescoring results where $N = 1000$

HTM wt.	HMM wt.	LM wt.	Accuracy/correct (%)
1	0	1	73.60/76.20
1	0	5	74.26/77.08
1	1	5	74.23/77.12
1	5	5	74.04/77.23
1	1	1	74.53/77.70
1	1.5	1	**74.59**/77.73

The HMM scores and bi-gram LM scores in the N-best lists are combined using the HTM scores with various weights. The HMM baseline gives accuracy of 73.04% with the optimal LM weight of 8.

Table 3
Lattice search results

HTM wt.	HMM wt.	LM wt.	IP	Accuracy/correct (%)
1	2	14	0	73.34/76.90
1	3	18	0	74.23/78.03
1	4	28	0	74.80/78.07
1	8	40	0	74.39/78.82
1	5	35	0	74.91/78.22
1	5	35	−0.5	74.99/78.25
1	5	35	−1	**75.02**/78.25

The HMM scores and bi-gram LM scores in the lattices are combined with the HTM scores using various weights. Phone insertion penalty (IP) is also varied. Warping factor is fixed at 0.480.

in the order of 10^{10}, and the averaged number of nodes expanded during the A* search is found to be between 20,000 (for short sentences) and 1,500,000 (for long sentences). The runtime CPU on a P-IV machine (decoder written in C#) for each test sentence is between 4 min (for short sentences) to 2 h (for long sentences), where most computing time is spent on computing log likelihood scores (especially when poor match occurs during search).

With adjustments of relative weights of various scores as well as the phone insertion penalty value, we obtained additional performance improvement, as shown in Table 3, over the best result in *N*-best rescoring.

We then empirically optimized the warping factor in the warped-LPCC version of the HTM, with detailed results shown in Fig. 7. The best accuracy achieved so far is 75.1% (i.e., PER = 24.9%). This performance is better than any HMM system as summarized in (Glass, 2003), and is approaching the best ever result for the same task (phone error rate of 24.4%) obtained by using many heterogeneous classifiers reported also in (Glass, 2003).

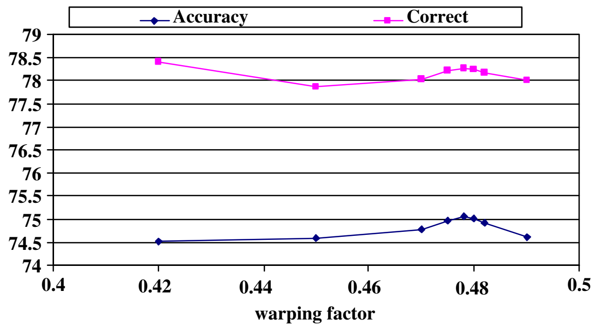

Fig. 7. Lattice search results. Performance as a function of the warping factor. The weight for the HTM scores, HMM score, and bi-gram LM score are fixed at 1, 5, and 35, respectively. Insertion penalty is fixed at −1. The best accuracy is 75.1% when the warping factor is 0.478.

6. Summary and conclusions

A long-contextual-span HTM presented in this paper captures underlying dynamic structure of speech coarticulation and reduction using a highly compact set of context-independent parameters. This is a significant extension of the earlier stochastic segmental models (e.g., Holmes and Russell, 1999; Ostendorf et al., 1996) and of earlier deterministic models with similar dynamics (e.g., Richards and Bridle, 1999; Bridle et al., 1998). This work also extends our earlier model implementation with two separate stages (Deng et al., 2006b) by combining them into an integrated system. The long-span nature of the HTM makes it difficult to develop efficient search algorithms for its full evaluation. The work presented in this paper is motivated by the desire to objectively evaluate a long-contextual-span HTM. Due to the high computational cost in implementing a full decoder, we used an *N*-best rescoring scheme as the first step in evaluating the model in the past. In this paper, we describe our recent work that extends our evaluation of the HTM from the earlier *N*-best to the current lattice-constrained decoding.

The special difficulty for the HTM decoding is that the HTM score for each frame needs to be determined by the model parameters associated with a variable number of adjacent phones, which in theory could reach the beginning and end of the utterance. The technical challenge due to this unusual long-contextual-span modeling is addressed in this work via a novel search technique based on carefully designed and implemented pruning strategies.

The search algorithm we have developed and presented is based on time-asynchronous A* search. Given the structural complexity of the long-span HTM, special algorithm designs presented in this paper include details on how the nodes and links in the lattices are expanded via look-ahead, how the A* heuristics are estimated, and what pruning strategies are applied to speed up the search. The experiments on TIMIT phonetic recognition show the capability of our newly developed lattice search algorithm in evaluating billions of hypotheses. This has been many orders of magnitudes higher than the limited number of hypotheses that could be evaluated by the HTM in our earlier *N*-best rescoring experiments.

Our experiment results on the TIMIT core test set show higher recognition accuracy than any recognizer without using many heterogeneous classifiers as reported in the literature. Our future work will involve further improvement of the HTM model design (e.g., exploiting differential input features) and more comprehensive evaluation of the model especially for larger recognition tasks with more casual speaking style than TIMIT. In the current work with the TIMIT data, we have observed only weak sensitivity of the stiffness parameter on recognition accuracy. The sensitivity is expected to be greater for casual speech, which will require automatic adaptation of the stiffness parameter in a phone-dependent manner. In addition, we will also investigate the possibility of generating novel acoustic–phonetic features from the HTM within a discriminative framework, permitting the use of non-uniform input features for recognizers.

References

Aubert, X.L., 2002. An overview of decoding techniques for large vocabulary continuous speech recognition. Comput. Speech Lang. 16, 89–114.

Bakis, R. 1991. Coarticulation modeling with continuous-state HMMs. In: Proc. IEEE Workshop Automatic Speech Recognition, Harriman, New York, pp. 20–21.

Bilmes, J., 2004. Graphical models and automatic speech recognition. In: Johnson, M., Ostendorf, M., Khudanpur, S., Rosenfeld, R. (Eds.), Mathematical Foundations of Speech and Language Processing. Springer-Verlag, New York, pp. 135–186.

Bridle, J., Deng, L., Picone, J., et al., 1998. An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition. Final Report for the 1998 Workshop on Language Engineering, Center for Language and Speech Processing at Johns Hopkins University, pp. 1–61.

Deng, L., 1998. A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition. Speech Comm. 24 (4), 299–323.

Deng, L., 2004. Switching dynamic system models for speech articulation and acoustics. In: Johnson, M., Ostendorf, M., Khudanpur, S., Rosenfeld, R. (Eds.), Mathematical Foundations of Speech and Language Processing. Springer-Verlag, New York, pp. 115–134.

Deng, L., Braam, D., 1994. Context-dependent Markov model structured by locus equations: Applications to phonetic classification. J. Acoust. Soc. Amer. 96 (October), 2008–2025.

Deng, L., O'Shaughnessy, D., 2003. Speech Processing – A Dynamic and Optimization-Oriented Approach. Marcel Dekker, New York.

Deng, L., Yu, D., Acero, A., 2004. A quantitative model for formant dynamics and contextually assimilated reduction in fluent speech. In: ICSLP 2004, Jeju, Korea.

Deng, L., Li, X., Yu, D., Acero, A., 2005. A hidden trajectory model with bi-directional target-filtering: Cascaded vs. integrated implementation for phonetic recognition. In: Proc. ICASSP, Philadelphia, PA, USA, pp. 337–340.

Deng, L., Yu, D., Acero, A., 2005. A long-contextual-span model of resonance dynamics for speech recognition: Parameter learning and recognizer evaluation. In: Proc. of IEEE Workshop on ASRU, 5 pages in CDROM.

Deng, L., Acero, A., Bazzi, I., 2006a. Tracking vocal tract resonances using a quantized nonlinear function embedded in a temporal constraint. IEEE Trans. Audio Speech Lang. Process. 14 (2), 425–434.

Deng, L., Yu, D., Acero, A., 2006b. A bi-directional target-filtering model of speech coarticulation and reduction: Two-stage implementation for phonetic recognition. IEEE Trans. Speech Audio Process. 14 (1), 256–265.

Gao, Y., Bakis, R., Huang, J., Zhang, B., 2000. Multistage coarticulation model combining articulatory, formant and cepstral features. In: Proc. ICSLP, Vol. 1, pp. 25–28.

Glass, J., 2003. A probabilistic framework for segment-based speech recognition. Comput. Speech Lang. 17, 137–152.

Goel, V., Byrne, W., 1999. Task dependent loss functions in speech recognition: A-star search over recognition lattices. In: Proc. EUROSPEECH 99.

Holmes, W., Russell, M., 1999. Probabilistic-trajectory segmental HMMs. Comput. Speech Lang. 13, 3–37.

Ma, J., Deng, L., 2003. Efficient decoding strategies for conversational speech recognition using a constrained nonlinear state-space model for vocal-tract-resonance dynamics. IEEE Trans. Speech Audio Proc. 11, 590–602.

Oppenheim, A., Johnson, D., 1972. Discrete representation of signals. Proc. IEEE 60 (6), 681–691.

Oppenheim, A., Schafer, R., Buck, J., 1999. Discrete-Time Signal Processing. second ed. Prentice Hall, Englewood Cliffs.

Ortmanns, S., Ney, H., 2000. Look-ahead techniques for fast beam search. Comput. Speech Lang. 14, 15–32.

Ostendorf, M., Digalakis, V., Rohlicek, J., 1996. From HMMs to segment models: A unified view of stochastic modeling for speech recognition. IEEE Trans. Speech Audio Proc. 4, 360–378.

Richards, H., Bridle, J., 1999. The HDM: A segmental hidden dynamic model of coarticulation. In: Proc. ICASSP, Vol. 1, Phoenix, pp. 357–360.

Rose, R., Schroeter, J., Sondhi, M., 1996. The potential role of speech production models in automatic speech recognition. J. Acoust. Soc. Amer. 99, 1699–1709.

Russell, S., Norvig, P., 1995. Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs.

Sun, J., Deng, L., 2002. An overlapping-feature based phonological model incorporating linguistic constraints: Applications to speech recognition. J. Acoust. Soc. Amer. 111 (2), 1086–1101.

Yu, D., Deng, L., Acero, A., 2005. Evaluation of a long-contextual-span trajectory model and phonetic recognizer using A^* lattice search. In: Proc. Interspeech, Lisbon, September 2005, pp. 553–556.

Zhou, J., Seide, F., Deng, L., 2003. Coarticulation modeling by embedding a target-directed hidden trajectory model into HMM. In: IEEE Proc. ICASSP, April 2003, Vol. I, pp. 744–747.