Evaluation of a Long-Contextual-Span Hidden Trajectory Model and Phonetic Recognizer Using A* Lattice Search

Dong Yu, Li Deng, Alex Acero

Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{dongyu, deng, alexac}@microsoft.com

Abstract

A long-contextual-span Hidden Trajectory Model (HTM) developed recently captures underlying dynamic structure of speech coarticulation and reduction using a highly compact set of context-independent parameters. However, the long-span nature of the HTM makes it difficult to develop efficient search algorithms for its full evaluation. In this paper, we describe our initial effort in meeting this challenge. The basic search algorithm is time-asynchronous A*. Given the structural complexity of the long-span HTM, special considerations are needed to take into account the fact that the HTM score for each frame depends on the model parameters associated with a variable number of adjacent phones. Specifically, we present 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 based on long-span HTM scores. The results significantly extend our earlier work from N-best rescoring to A* search over lattices.

1. Introduction

Modeling long-span contextual effects in speech acoustics is important for spontaneous speech recognition, where phonetic reduction and coarticulation are often mixed in causing tremendous variability in the speech signal. One particular type of speech models recently developed [1][2] exploits temporally bi-directional (forward and backward) filtering of vocal tract resonance (VTR) targets to achieve such long-contextual-span modeling. The filtered VTR trajectories are treated as the hidden vectors and a nonlinear prediction with statistical residuals generates the cepstral parameters from the VTR trajectories. The statistical characterization of this hidden trajectory model (HTM) permits straightforward computation of the model likelihood score for the cepstral observation data given the phone sequence and phone segment boundaries. The technique of parametric filtering on VTR as part of structured speech dynamics permits the use of a compact set of context-independent parameters to represent long-span contextual effects in the acoustic model. This distinguishes HTM from other types of acoustic models such as HMMs.

Initial evaluation of the HTM on TIMIT database with an N-best paradigm was carried out using a phonetic recognizer built with the HTM, as reported in [2]. Recent improvement on the model design and new training are reported in [3], with substantial performance improvement in the N-best rescoring result. The research reported in this paper extends the evaluation from the N-best rescoring paradigm to more rigorous lattice search.

Our previous work on N-best rescoring is limited in the richness of the lists in providing sufficient phonetic sequence information to evaluate the HTM. We used a high-quality HMM system (HTK-based) to produce N=1000 lists for each test utterance and found that the average oracle phone error rate is as high as 18% for the TIMIT core test set. The long-contextual-span nature of the HTM presents a special challenge of long-contextual-span modeling in our HTM --- the error-spreading effect. That is, when phone errors are present in the N-best list, they tend to spread over several adjacent phones. This is caused by the VTR filtering where the adjacent phones' VTR trajectories would be strongly influenced by the incorrect VTR targets associated with the incorrect phones in the N-best list. In our earlier work [2] , we artificially removed the error-spreading effect by appending the reference hypothesis into the N=1000 list and we generally observed a much higher HTM likelihood scored on the reference (free from the error-spreading effect) than on virtually all remaining 1000 hypotheses. This demonstrates a highly desirable property of HTM. This property is necessary for HTM to beat HMM, but is not sufficient.

We have not found it beneficial in using HMM to generate much larger N-best lists for evaluating the HTM. For example, when N=2000 best lists are generated, we found that the oracle error rate is only 0.8% lower than that for the N=1000 best lists. Even with the large phone lattices built with the HMM system, we still found substantial oracle errors present. However, lattices offer much richer hypotheses than N-best lists, facilitating more rigorous evaluation of the HTM. This forms the key motivation of this work.

2. Algorithm for Lattice Search

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.) 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 phone lattice.

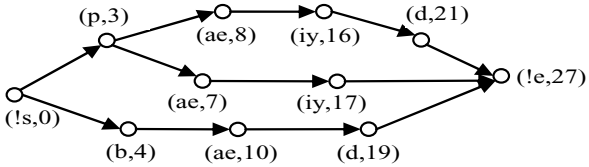

Fig. 1: An example of a phonetic lattice

2.1. 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 [4]. In our implementation, the search over the lattice is conducted using the one-stack A* search algorithm described in Fig. 2.

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. 2: The basic one-stack A search algorithm*

. It has been well known [5] that A* search algorithm can always find the best hypothesis 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.

2.2. Score Estimation

As indicated in Section 2.1, the performance of the A* search algorithm depends heavily on the quality of the estimated 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 (1):

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

where S_i is the log-likelihood of the *i*-th 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:

$$\hat{S} = S_p + H_n \tag{2}$$

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's obvious that the quality of the score estimation using (2) depends on the quality of the heuristic score H_n.

To make sure the backward heuristic score is a good estimate of the true backward score, we developed the following score computation algorithm. 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 H_n in (2). It also suggests that we may estimate the true score by using the same formula 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 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. 3, 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 lowering of the score caused by this approximation, we adjust the heuristic score by adding a fixed, small score for each frame so that the heuristic score will highly likely satisfy Property 2.

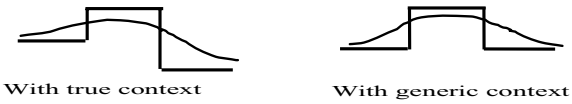

Fig. 3: VTR Trajectory with and without Context

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. Fig. 4 describes the algorithm to calculating the backward heuristic scores H_n . The calculation of H_a and H_n incurs very small computational cost.

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. 4: Computing backward heuristic scores of nodes H_n

2.3. Hypothesis Expansion

One of the most important steps in A* search is expansion of the hypotheses. If the score of each phone (arc) were independent of the context, the expansion would be trivial, and we only need to follow all the outgoing arcs of the final node in the current partial hypothesis to obtain the list of expanded partial hypotheses. However, 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 can not simply use the nodes in the original lattice for the hypothesis expansion. 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 in n_p and n_f 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:

$\{(!s,0),(p,3),(ae,8)\}$ and

$\{(!s,0),(p,3),(ae,7)\circ(iy,17)\}$.

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 infers 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. Given this observation, we can expand a partial hypothesis by directly traversing the original lattice according to the algorithm described in Fig. 5. 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

$$S_{p,new} = S_{p,hyp} + S_a \quad (3)$$

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. 5: 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. ∘ denotes concatenation.

3. Pruning Strategies

The lattice rescoring can be very slow and it consumes much memory given a large number of possible expansions in the HTM. To speed up the rescoring process, we implemented several pruning strategies.

3.1. 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. 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-calculation. The hash value used in our system is derived from the VSCD-node's fingerprint, which is consisted of the 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 4/5 of the calculation.

3.2. Prefix Pruning

In the A* search process, we only keep these 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. This pruning strategy is called prefix pruning since the hypotheses pruned out have the same prefix phone list.

3.3. Pruning on Unseen Phone Bi-gram

The basic idea behind the unseen phone bi-gram pruning is that the correct hypothesis should not contain a large percentage of unseen phone bi-grams. This type of pruning eliminates the score calculation for the hypotheses after the percentage of unseen bi-grams associated with them has exceeded a specific threshold (e.g., 10%). The bi-gram list is obtained from the training set. The larger the threshold, the less effect the pruning has, and the higher chance that the best hypothesis will not be pruned out. This threshold is tuned based on a held-out set

3.4. Pruning on Heuristic Score

We have observed that the heuristic score of the true hypothesis is generally not very far away from the best possible heuristic scores 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.

3.5. Joint Node Pruning

If several partial hypotheses join at the same VSCD-node, then we only need to keep the hypothesis with the highest partial score. This pruning 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.

4. Experimental Results

We have conducted phonetic recognition experiments on the TIMIT standard core test set, aimed to evaluate the phonetic recognizer built on the basis of the long-contextual-span HTM. We compare and analyze the results obtained with N-best and lattice-search-based rescoring. The phone lattice used in our experiments is 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.

Results of the lattice search algorithm described in Sections 2 and 3 on phonetic recognition are presented in Table 1, where the sensitivity to phone bi-gram language model weights (LM) and phone insertion penalties (IP) is shown. 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 a large percentage of oracle errors. To artificially remove the "error spreading effect", we inserted the references into the very large initial lattices.

After the various stages of pruning, the averaged total number of hypotheses evaluated is estimated to be 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 hrs (for long sentences).

The results shown in Table 1 demonstrate consistent trends in our earlier work [2][3] that HTM outperforms HMM (accuracy 73%) under the same experimental condition of acoustic information and language models and with the error spreading effect removed.

Table 1: TIMIT phonetic recognition results using the HTM to rescore large lattices. No HMM scores are used.

	LM=8;IP=-0.5	LM=6;IP=-0.5	LM=8;IP=0
Corr	91.12% (6682)	90.59% (6643)	90.82% (6660)
Sub	6.67% (489)	7.38% (541)	6.98% (512)
Del	2.21% (162)	2.03% (149)	2.20% (161)
Ins	4.66% (342)	5.26% (386)	4.91% (360)
Acc	86.46%	85.33%	85.91%

5. Summary and Conclusions

This work is motivated by the desire to objectively evaluate a long-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 on extending our evaluation of the HTM from N-best to lattice rescoring

The companion paper [3] describes the HTM design, training, and score computation. In particular, the HTM score for each frame is 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 challenge due to such unique long-contextual-span modeling is addressed in this work via A* search, using carefully designed and implemented pruning strategies.

Our experimental results demonstrated that the HTM generally scores higher on the correct hypothesis than billions of competing incorrect hypotheses. Our future work will aim at translating this success to a realistic recognizer where correct hypotheses (and their minor variants) can be directly found by the HTM.

6. References

[1] L. Deng, D. Yu, A. Acero, "*A quantitative model for formant dynamics & contextually assimilated reduction in fluent speech*", Proc. ICSLP, pp 719-722, Jeju, Korea,2004.

[2] L. Deng, X. Li, D. Yu, & A. Acero, "*A Hidden Trajectory Model with Bi-directional Target-Filtering: Cascaded vs. Integrated Implementation for Phonetic Recognition*", Proc. ICASSP, pp 337-340, 2005, Philadelphia, PA, USA.

[3] L. Deng, D. Yu, A. Acero, "*Learning statistically characterized resonance targets in a hidden trajectory model of speech coarticulation and reduction",* submitted to INTERSPEECH, 2005.

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

[5] S. Russell, and P. Norvig, "*Artificial Intelligence: A Modern Approach*", Prentice Hall, Englewood Cliffs, 1995.