A TIME-SYNCHRONOUS PHONETIC DECODER FOR A LONG-CONTEXTUAL-SPAN HIDDEN TRAJECTORY MODEL

Xiaolong Li, Li Deng, Dong Yu and Alex Acero

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

ABSTRACT

A novel time-synchronous decoder, designed specifically for a Hidden Trajectory Model (HTM) whose likelihood score computation depends on long-span phonetic contexts, is presented. HTM is a recently developed acoustic model aimed to capture the underlying dynamic structure of speech coarticulation and reduction using a compact set of parameters. The long-span nature of the HTM had posed a great technical challenge for developing efficient search algorithms for full evaluation of the model. Taking on the challenge, the decoding algorithm is developed to deal effectively with the exponentially increased search space by HTM-specific techniques for hypothesis representation, word-ending recombination, and hypothesis pruning. Experimental results obtained on the TIMIT phonetic recognition task are reported, extending our earlier HTM evaluation paradigms based on N-best and A* lattice rescoring.

Index Terms: Hidden Trajectory Model, time-synchronous decoding, trace-based hypothesis, TIMIT

1. INTRODUCTION

With similar motivations to the development of a structured language model where syntactic structure is exploited to represent long-distance relations among words in sentences [1], we recently developed a version of the *structured speech model* in which dynamic structure of speech is exploited to characterize long-span contextual influence among phonetic units in fluent speech utterances [2,3]. The particular version of the structured speech model developed, which we call *hidden trajectory model* (HTM), represents the speech structure in the unobserved (hidden) vocal tract resonance (VTR) domain, and an analytical nonlinear function is provided to map from the time-varying VTR vector to the observed cepstral vector. Unlike the Hidden Markov Model (HMM) where (short-span) context dependency is achieved by non-parametrically enumerating adjacent phonetic contexts (e.g., triphone), our HTM employs parametric temporal filtering of VTR targets as the basis for the characterization of long-span coarticulation. The filtered VTR trajectories are treated as the hidden vector sequence, and a nonlinear prediction with statistical residuals, which are represented by a context-independent single-Gaussian HMM, generates the cepstral vectors. The statistical characterization of this HTM allows straightforward computation of the model likelihood score for the cepstral observation data if the phone sequence and phone segment boundaries are given.

Previous evaluation of the HTM were based on N-best list rescoring [2], where each of the hypothesized phone sequences and their segment boundaries are known. More recent evaluation extended the N-best rescoring paradigm to A*-based lattice rescoring where much richer hypotheses were made available for evaluation [6]. The research reported in this paper further extends the evaluation from the N-best/lattice rescoring paradigms to the more flexible time-synchronous search paradigm, and is aimed to equip HTM with a full decoding capability.

Because N-best or lattice rescoring is limited in the fixed phone boundaries given by the initial decoding results based on HMM, it has not been able to exploit the full power of the HTM. In analyzing the error patterns in the phonetic recognition results from N-best rescoring, we found that when the phone boundaries were selectively shifted by up to 4 frames for the correct "oracle" hypotheses, these hypotheses tend to give significantly higher scores than most other competing hypotheses with and without boundary shifts. This points to the possibility that the segment boundaries given by the HMM from either the N-best lists or lattices are not "optimal" and hence not adequate for HTM rescoring. This also suggests that a time-synchronous decoder for the HTM is highly desirable because it automatically determines the optimal segment boundaries for each phone in the sentence while searching for the optimal phone sequence.

Currently the most successful time-synchronous decoding algorithm for continuous speech recognition is based on Dynamic Programming (DP) [4], where the best word sequence is found by iteratively extending partial word-hypotheses frame-by-frame and all competing word-hypotheses can be effectively pruned based on the same acoustic observations. When trying to borrow the same idea to carry out time-synchronous decoding for the HTM, however, serious challenges would arise in hypothesis representation and pruning because of the long-span nature of the HTM leading to a fast, exponential increase in the search space. Many modifications to the classic DP-decoding paradigm are necessary to make the search space manageable, which we will address in detail in this paper.

This paper is organized as follows. Section 2 gives the basic framework for HTM-based time-synchronous decoding. Section 3 presents several specific decoding challenges associated with the long-span nature of the HTM and our current solutions. We emphasize the key differences from the problems in conventional, HMM-based time-synchronous decoder design. Section 4 discusses detailed data structures employed for the decoder's development. Section 5 presents experimental evaluation results.

2. DECODING ALGORITHM OUTLINE

The basic structure of the search algorithm developed in this work is provided in Figure 1, with three levels of search hypotheses---

word, phone, and residual state.[1] The long-span nature of HTM requires that phone-level hypotheses are represented by a special data structure, which we call "trace" (see details in Section 3). And to simplify the design of the decoder, we adopt a linear lexicon instead of a lexical tree for the search organization.

As seen in Fig. 1, the general structure of the HTM decoding structure is similar to the classic DP-based algorithm for HMM [4,5]. At each frame during decoding, there are similar sub-routines for 1) hypothesis-extension; 2) hypothesis-pruning and 3) word-ending recombination/pruning (including recording word-hypotheses into back-tracing array). And at the last frame of a sentence, both HTM and HMM decoders perform back-tracing and output the recognized word sequence from the back-tracing array. However, as will be presented in the next section, critical differences between the HTM and HMM decoding algorithms are in the details of the above three subroutines. And in many places new solutions are necessary to deal with the long-time spanning of the HTM traces. In addition, a new subroutine, called "Merge the same next-time traces" (not needed in the HMM decoder) in Fig. 1, is carried out before extending new hypotheses so as to reduce the number of active traces. This makes the search space as compact as possible.

```
For each sentence in the testing set
  Initialization for t = 0
  For t = 1... Maximal_Frame_Number-1
    Merge the same next-time traces
    Extend new traces/hypotheses by DP
    Prune all state-level hypotheses and traces
    Perform word end recombination and pruning
  End For
  Back-tracing and outputting word sequence
End For
```

Fig.1. Basic time-synchronous search algorithm for HTM

3. SPECIFIC DECODING ISSUES RELATED TO HTM

3.1. Representation of Hypotheses

In the HMM-based decoding, the likelihood of each frame (usually 10ms of speech) given the HMM state depends only on the acoustic observation of the current frame, making the representation of search hypotheses extremely simple. In contrast, in the HTM model, the likelihood of each frame (for the given residual state in a phone) depends not only on the observation and phone identity of current frame, but also on the phone identities associated with both previous and future D (D>0) frames (see [2] for a detailed mathematical description of such long-span dependency). In the current HTM model, we set D = 7. That is, each hypothesis has to record all the phone identities for a 15-frame-long window centered at the current frame, and any difference in any phone identity within this (2D+1)-frame window will cause a difference of the acoustic likelihood score. In our current solution to representing this complex "variable-gram" dependency, each search hypothesis is made associated with a special data structure called "trace". (Note here "trace" is obviously different from "Word Trace" concept in [7], which was used to represent a word-level lattice based on traditional HMM.) Each trace can be considered as a "super-state" and it includes a 15-frame-long array that records the identities of all the phones within this time window. Further, associated with each trace there is a pointer to a small array that represents the active residual state(s) for the phone hypothesis. Figure 2 illustrates these two-level hypothesis presentation used in the HTM decoder. It shows the changes of phone identities when a trace proceeds from time frame t = 7 into t = 9. Each residual model has 3 emitting states and 2 non-emitting states, and the size of the active state-level hypotheses attached to each trace is dependent on the position of current phone (the center, shaded block of the window).

Similar to the classic DP-based algorithm, all different traces, residual states, and word hypotheses activated at each frame are organized in arrays which enable direct access to any level of the search hypotheses.

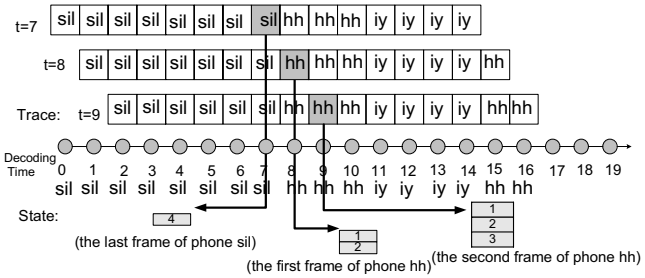

Fig. 2. Trace and state-level hypotheses in the HTM decoder

3.2. Handling Exponentially Increasing Search Space

One critical difference between the time-synchronous decoding algorithms for HTM and for HMM is the size of the search space. For HMM, the search space grows essentially linearly with the length of the sentence. The linearity is due to the HMM property that the frame likelihoods given the state are independent of each other, which makes it possible to effectively employ word-ending recombination to reduce the search space. In contrast, in HTM, this conditional independence assumption no longer holds, and the long-span contextual influence makes the number of active traces grow exponentially. It is very difficult, if not impossible, to manage this large search space using regular DP methods [4].

To make the search space reduced and compact, we have developed and applied two techniques. First, the lattices generated using the regular triphone HMM system are used to constrain time-synchronous search using the HTM. Starting from the regular lattices generated from HTK, we do post-processing on these lattices by dropping time information at each node and merging the nodes that differ only in timing information. This procedure is very similar to the minimization processing of Weighted Finite State Transducer (WFST), and the resulting lattice becomes only 1/3 to 1/2 in the size compared with the original lattice. With the compressed lattice, the extension of new traces will be constrained by the permitted path in the lattice, significantly reducing the size of search space to one with a manageable order of magnitude.

Second, approximate word-ending recombination is exploited to reduce the search space. This technique is illustrated in Figure 3, we assume three different paths reaching the same word ending of w at time $t = t_1$, and $P_3<P_2<P_1$, where P_i is the path score for path i. If the decoding were based on HMM with the within-word acoustic model and bigram language model, then word recombination

[1] The residuals are the feature vectors that cannot be predicted by the HTM, and are represented by a three-state HMM with a regular left-right topology; see details in [2].

would allow keeping only path 1, and paths 2 and 3 can be discarded without loss of search accuracy. However, in the HTM-based decoding, such word recombination will produce search errors because path 2 and path 3 may gain a higher score than path 1 in a near-future frame resulting from the 15-frame-long contextual window. To reduce the possibility of this search error, approximate word recombination is developed and adopted as follows: 1) pre-set a recombination threshold as T_w, and obtain the highest path score as P_{best} among all those paths reaching w at time t_1; 2) prune those paths if their scores are $P_i < P_{best} - T_w$. In Figure 3, assume $P_3<P_1$-5.0, $P_2>P_1$-5.0, and the recombination threshold is T_w=5.0. Then only path 3 will be pruned and paths 1 and 2 will be kept for further extension.

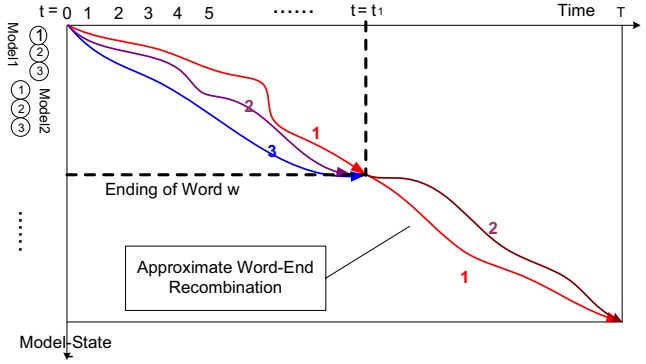

Fig.3. Approximate word-ending recombination. Assume path scores for paths 1, 2, 3 satisfy: $P_3<P_2<P_1$, and $P_3<P_1$-5.0, $P_2>P_1$-5.0. (and assume 5.0 as the recombination threshold).

3.3. Frame-Incremental Extension of New Traces

Figure 4 gives the algorithm for extending new traces at each frame of decoding. For each old trace in the trace list, the extension of new traces depends on the previous phone's identity within the 15-frame-long window of the old trace.

```
For all active traces in trace list
  Get all new traces by extending the previous phone's ID of old trace
  For all possible new traces
    Calculate HTM residual vectors
    For all active state-level hypotheses of old trace
      Calculate HTM state likelihood
      (Calculate HMM state likelihood; obtain combined score)[2]
      Perform state-level extension and recombination by DP
      Save the new hypotheses into the state hypotheses list
    End For
    Save those new traces into trace list
  End For
End For
```

Fig. 4. Algorithm for extending new traces by a new frame

For the main loop in Figure 4, the major computational cost lies in computing HTM residual vectors and computing the HTM state likelihood. For detailed procedures of the computation, readers are referred to [2,3].

3.4. Pruning Methods

As with classic DP-based time-synchronous HMM decoding, good pruning methods are highly desirable to make the search space compact while keeping the best path away from being pruned. Two different pruning methods are used in HTM decoder, beam pruning [4] and histogram pruning [5], for all state, phone and word levels. Hence, in total, there are six tunable pruning parameters --- state beam, state histogram threshold, phone beam, phone histogram threshold, word beam (or LM pruning beam), word histogram threshold (or LM cut-off threshold). The three beam parameters are expressed as the likelihood scores, and the three histogram threshold parameters are expressed as the numbers of candidates to be kept.

Special considerations are needed for setting the histogram threshold pruning parameters for the HTM. We have observed that the likelihood values are very close to each other among the many traces that differ in only one phone identity among their 15-frame-long phone identities. Hence, if we were to set the histogram threshold parameter N as the permitted number of highest scored candidate paths, N would need to be extremely large (hundreds of thousands) in order not to lose the correct hypothesis. This would make the pruning operations highly inefficient and would take unrealistically large memory. We have devised an alternative technique to successfully circumvent these problems. The technique considers only the phones that are near the center of the 15-frame-long phone identity array (i.e., near the "current" phone). That is, we ignore the phones in the two far ends of the array which have a minimum effect in the trace's likelihood. In implementing this technique, we attach each HTM trace with a triphone ID to represent the nearest left- and right-contexts of the current phone. Then, only the traces with different triphone IDs are counted as different traces for pruning threshold setting of N.

4. DATA STRCTURES

Similar to the DP algorithm used for HMM decoding [4], one major data structure used in HTM decoding is *list* (or *array*). There are four lists used in the HTM decoder to store the active hypotheses for traces, states, word-endings, and words in back-tracing array, respectively. Figure 5 illustrates an example of the trace list and state list used in the HTM decoder. We set a pointer in each trace pointing to the ending position of the state array for the current trace. This makes it very easy to directly access any hypothesis. Figure 5 also shows some class members (defined in Visual C# language) that represent trace- and state-level hypotheses. As an example, a one-byte-type, 15-element array called "code" in class Trace, is used to store 15-element phone identities of each trace. Among other members of class Trace, there is a "c_link" member as the current link of the lattice, and an "n_link" member as the next link of the lattice for the current trace. Both of them are objects of class "LatLink". Finally, there is a string-type member "t_hmm" to represent the triphone ID corresponding to current trace.

In addition to the list, the data structure *hash table* is also used extensively in designing our HTM decoder, and is responsible for implementing the functionalities of pruning and recombination. Because of the simplicity of using hash table in Visual C# (class "Hashtable"), the pruning and recombination of different search hypotheses are implemented with high efficiency. Also, one of key subroutines, "Merge the same next-time traces", discussed in Section 2 and shown in Figure 1, is implemented efficiently and conveniently by hash table. Specifically, the active traces activated

[2] This is an optional step. We found in the experiments that the use of the combined HMM score significantly speeds up the search and improves the recognition accuracy.

in the immediately previous time is merged when they have the same "next-time" phone identities (i.e., the same in all "code" class members except for code[0]), because they will be extended to the unique new trace.

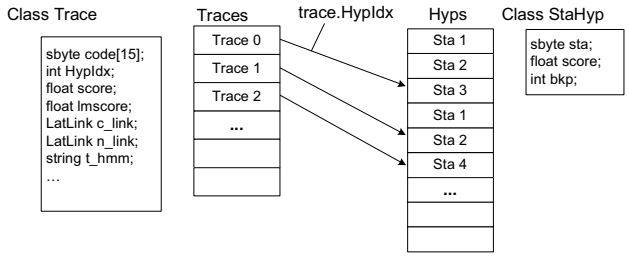

Fig.5. Lists and class members used for traces and state-level hypotheses (in Visual C#).

5. EVALUATION EXPERIMENTS

In this section, experimental results on the standard TIMIT phonetic recognition task are reported. The HTM model parameters, including VTR target means and variances and cepstral-prediction residual parameters, are trained from the TIMIT training set as described in [2]. The test set is the standard core test set including 192 sentences consisting of 7333 phone-like units. The time-synchronous HTM decoder is constrained by lattices computed by HTK with crossword triphone HMMs, which has a baseline phone accuracy rate of 72.50%. The oracle accuracy for the lattices is 97.41%. The accuracy of A* based lattice rescoring using HTM model only is 74.3%, and when HTM combines with HMM for A* lattice rescoring, the accuracy reaches 75.07% [3].

Table 1. Phonetic recognition accuracy and relative HTM decoding CPU time/utterance as a function of the decoder's pruning parameter settings

Beam-Pruning Thresholds	Histogram-Pruning Thresholds	Relative Time	Acc%
10.0, 10.0, 10.0	50, 50, 20	1.0	17.22
50.0, 50.0, 50.0	50, 50, 50	8.0	71.46
100.0, 100.0, 100.0	500, 500, 200	10.2	73.57
1000.0, 1000.0, 500.0	20000, 20000, 800	36.2	73.97
3000.0, 3000.0, 1000.0	20000, 20000, 800	78.1	74.43
3000.0, 3000.0, 1000.0	30000, 30000, 2000	138.1	74.68

Table 1 summarizes a set of experimental results where the phonetic recognition accuracy and the relative HTM decoding CPU time (time was measured for a given TIMIT utterance) is shown as a function of the six tunable decoder pruning parameters as discussed earlier in the paper. These parameters include three beam pruning thresholds (state-, phone-, and word-level[3]), and three histogram pruning thresholds. The approximate word-ending recombination threshold is set as 10. The weights for the log-HMM-likelihood, the log-HTM-likelihood, and the log-LM-probability are set to be 5, 1, and 20, respectively. And the phone insertion penalty is set to be -40. We observe that as the threshold values are increasing and the fewer and fewer intermediate hypotheses are pruned, the recognition accuracy is progressively increasing but at the expense of an increasingly greater computational cost.

[3] For the phonetic recognition task, the word-level pruning differs from the phone-level pruning only in its use of the (bi-gram or bi-phone) LM. Our decoder software, however, has been written to handle general word lexicons.

6. SUMMARY AND CONCLUSION

In this paper, a time-synchronous decoder designed for the long-contextual-span HTM as a special type of structured speech model is presented, which extends our previous work of HTM evaluation based on N-best list rescoring and lattice rescoring. The long-span nature of the HTM poses serious challenges in hypotheses management and in efficient pruning/recombination methods for the DP-based decoding algorithm. Novel approaches developed to address these challenges include the trace-based contextual-phone hypothesis representation, the approximate word-ending recombination technique, and the histogram-based hypothesis pruning method. These innovations, together with the use of lattices as a constraint for the search, have successfully reduced the exponentially increased search space to a manageable, moderately-sized space. The experimental results on a standard TIMIT phonetic recognition task demonstrate the effectiveness of the decoding algorithm, especially the new techniques developed within the algorithm to handle the otherwise unmanageably large search space. While the phonetic recognition accuracy achieved by the current decoder is already higher than most techniques (including HMM techniques) published in the literature, further tuning of the decoder parameters, as well as improved HTM design and training, are expected to further improve the accuracy. Finally, we have been in the process of completing the development of the decoder for larger tasks of continuous speech recognition with more realistic "lexicon" than the trivial one in the phonetic recognition task presented in this paper. Our future work will involve some larger scale evaluation of the HTM.

6. ACKNOWLEDGEMENT

We thank Patrick Nguyen for lattice minimization tool and useful discussions, and thank Asela Gunawardana and Mike Seltzer for generating high-quality lattices from an HMM system for our use.

7. REFERENCES

[1] C. Chelba and F. Jelinek. "Structured Language Modeling," Computer Speech and Language, October 2000, pp. 283-332.
[2] L. Deng, X. Li, D. Yu and A. Acero, "A Hidden Trajectory Model with Bi-Directional Target-Filtering:" Proc. ICASSP, pp 337-340, 2005.
[3] L. Deng, D. Yu, X. L. Li, and A. Acero, "A Long-Contextual-Span Model of Resonance Dynamics for Speech Recognition: Parameter Learning and Recognizer Evaluation," IEEE Workshop on Automatic Speech Recognition & Understanding, Nov. 2005.
[4] H. Ney, S. Ortmanns, "Dynamic programming search for continuous speech recognition," IEEE Signal Proc. Magazine, 16, pp.64-83, 1999.
[5] A. Sixtus, Crossword Phoneme Models for Large Vocabulary Continuous Speech Recognition. Ph.D. dissertation, RWTH, Germany, 2003.
[6] D. Yu, L. Deng, and A. Acero, "Evaluation of a Long-contextual-span Hidden Trajectory Model and Phonetic Recognizer Using A* Lattice Search," Proc. of Eurospeech, Lisboa, Sep. 2005.
[7] G. Zweig and M. padmanabhan, "Exact Alpha-Beta Computation in Logarithmic Space with Application to MAP Word Graph Construction," Proc. of ICSLP, Beijing, China, 2000.