A PUBLIC DOMAIN DECODER FOR LARGE VOCABULARY CONVERSATIONAL SPEECH RECOGNITION

Neeraj Deshmukh, Aravind Ganapathiraju, Jonathan Hamaker, Joseph Picone

Institute for Signal and Information Processing
Department for Electrical and Computer Engineering
Mississippi State University, Mississippi State, MS 39762
{deshmukh, ganapath, hamaker, picone}@isip.msstate.edu

ABSTRACT

The high cost of the infrastructure required to conduct state-of-the-art speech recognition research prevents many small research groups from evaluating new ideas on large-scale tasks. To overcome this barrier, we are developing an Internet-based speech-to-text (STT) toolkit. In this paper, we present the core component of this system: a decoder that uses a one-pass time-synchronous Viterbi-based search algorithm called trace projection. This decoder can support efficient lattice rescoring using cross-word triphones, lexical trees and n-gram grammars. The decoder performance in terms of CPU and memory usage is on par with commercial systems of its kind. Preliminary evaluations on the SWITCHBOARD (SWB) corpus have yielded a word error rate of 39%.

1. INTRODUCTION

A speech-to-text (STT) system conceptually consists of three subsystems — an *acoustic processor* which converts the speech signal into a sequence of feature vectors modeled using Hidden Markov Models (HMMs); a *linguistic processor* which predicts the next word given a sequence of previously recognized words; and a *search engine* which searches a large word graph to find the most probable word sequence given the observed acoustic evidence. We refer to the latter as a *decoder*, and represent it using Bayesian statistical inference —

$$P(W_t^i \left| \boldsymbol{O}_t \right.) = (P(\boldsymbol{O}_t \left| W_t^i \right.) P(W_t^i)) / P(\boldsymbol{O}_t) \quad (1)$$

The number of possible word sequences is quite large even for a small vocabulary. For a state-of-the-art system using a trigram language model (LM) and cross-word triphones, the path calculations involve a hierarchy of graphs (sentences, words, phones, and HMM states). The control structure required to perform this search efficiently is extremely hard to implement. Hence very few good decoders exist, and the best decoders are always considered proprietary.

The requisite investment required to indigenously develop or license quality software is prohibitively high. As an upshot, the participation in state-of-the-art speech research is decreasing at a time when the technology has generated unprecedented world-wide interest. One way to decrease the overall cost of STT research is to use the Internet as a means to share resources and provide access to the fundamental technology. The public domain decoder developed by the Institute for Signal and Information Processing (ISIP) is a step in that direction.

2. THE ISIP DECODER

A state-of-the-art public domain decoder needs to efficiently and transparently handle tasks of varied complexity, from connected digits to spontaneous conversations. Therefore the current release of the ISIP decoder is equipped to handle decoding of cross-word triphones and n-gram language models. It handles multiple pronunciations of words and large lexicons easily through dynamically constructed lexical pronunciation trees. It can also efficiently rescore lattices generated using a previous search.

2.1. Trace Projection

We have developed a variation of the standard Viterbi-style time-synchronous search paradigm [1] called *trace projection* for the decoder. The primary data structure used to propagate path information along the frames is called a *Trace*. The decoder uses two sets of traces — phone level and state level. Each

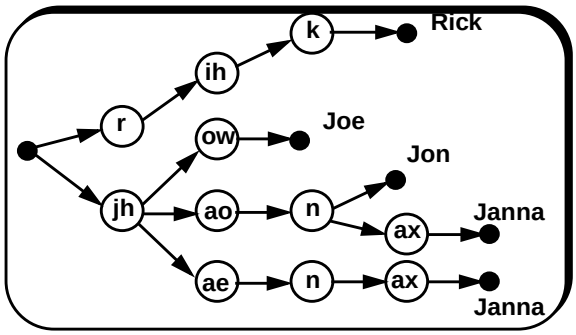

Figure 1: Lexical tree structure used with the ISIP decoder — triphones such as *jh-ao+n* are generated dynamically and shared by both words *Jon* and *Janna*.

trace contains an index to the current lattice node, lexical tree node and the triphone model, thus providing a hierarchical representation for each instance in the search space. It also maintains the temporal path history and score.

At each instantiation of a triphone, a state-level trace is projected from the previous phone-level trace and added to a state-level trace list. For each frame, the active states are evaluated only once. The state-level traces are compared and the best trace for each different instance of the state is projected to the next states as governed by the state transition probabilities. The score for each state is stored locally and added to the projected trace path score. A trace exiting the HMM is added to the phone-level trace list and used to project the next triphone traces.

2.2. Lattice Compaction

During lattice rescoring, if we ignore the timing information associated with the lattice nodes, many arcs of the lattice indicate essentially the same word sequence and need not be decoded individually [2]. The original lattice is converted into a word graph that preserves all the word hypotheses, yet merges all such duplicate arcs. This achieves reduction in the effective lattice size by a factor of 2 to 5 and causes a significant drop in the search space complexity at minimal computational overhead.

2.3. Lexical Trees and Triphone Generation

The ISIP decoder uses lexical trees to represent the pronunciations of all the words following a particular node in the lattice, as well as for efficient LM look-ahead [3]. Each lexical tree node is associated with a monophone in the word pronunciation (see Figure 1). Triphones are generated dynamically from these nodes at each step. This reduces the tree size and facilitates creation of triphones on an as needed basis. A lexical tree is created only when the predecessor lattice node is reached in the decoding process. It is shared by multiple instances of that lattice node. A lexical tree no longer used for decoding is pruned away to save on memory.

2.4. Path Merging

The decoder can share the evaluation of similar parts of different hypotheses, and therefore prevent the computation from increasing exponentially with the temporal expansion of the search space. This can be achieved by merging hypotheses that intersect in the search space at points wherefore their future is identical, such as word ends. Here only the highest-scoring trace is propagated for each triphone instance at the end of a particular word. The lattice and lexical tree structure of the decoder framework automatically ensures that all the partial hypotheses represented by these traces have identical linguistic contexts.

2.5. Pruning

The ISIP decoder employs three different heuristic pruning techniques to prevent growth of low-scoring hypotheses, and thus reduces the computational load.

Global beam pruning: The global beam is determined empirically, and applied to all traces at the phone and state level. At a time t, the best scoring hypothesis in state s has a path score given by

$$q_{max}(s, t) = max\{q(s, t)\} \ldots \forall s . \quad (2)$$

Then for a global beam width of $b(t)$, we prune all hypothesis (s, t) such that

$$q(s, t) < q_{max}(s, t) + b(t) . \quad (3)$$

Word-end pruning: Traces signifying the end of words are further pruned to curb the fan-out caused by the LM list of possible next words. The best word-ending hypothesis score at a time t for a word w is

$$Q_{max}(w, t) = max\{Q(w, t)\} \ldots \forall w . \quad (4)$$

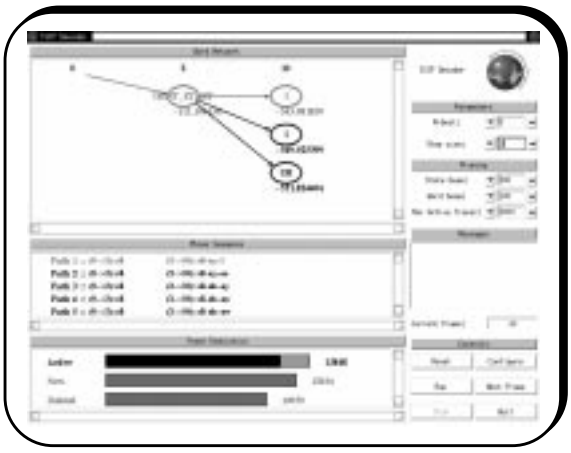

Figure 2: Screenshot of the decoder graphical interface.

If the word-end beam width is set to $B(t)$, then all hypotheses (w, t) satisfying the following criterion are pruned away —

$$Q(w, t) < Q_{max}(w, t) + B(t). \quad (5)$$

Maximum active traces (MAT) pruning: By setting an upper limit on the number of active triphones per frame we can effectively regulate the memory usage (and hence computation time) of the decoder [4]. If the number of active hypotheses exceeds this limit *max_num*, then only the best *max_num* hypotheses are allowed to continue while the rest are pruned off.

3. SOFTWARE AND INTERFACE DESIGN

The ISIP decoder is designed in an object-oriented fashion and written completely in C++. For efficient access and sorting purposes the principal data structures are handled via linked lists and hash tables. Efficient modules for memory management ensure that used memory is periodically freed and reused. The software structure allows for a hierarchical representation of the search space extensible to higher levels such as sentences. Hooks are provided to apply various kinds of acoustic distributions.

The distribution of the decoder includes a Tcl-Tk based graphical interface (GUI) that allows the user to specify various decoding parameters through a simple point-and-click mechanism (Figure 2). It also provides a frame-by-frame display of the top word hypotheses, triphones and memory usage statistics; thus serving as a debugging and educational tool.

Test data	ISIP decoder		Baseline decoder	
	time	memory	time	memory
AD1	41.2 s	46 MB	111.9 s	26 MB
AD2	132.2 s	53 MB	435.8 s	47 MB
AD3	57.9 s	46 MB	224.9 s	30 MB
SWB1	136.6 s	64 MB	145.7 s	61 MB
SWB2	67.9 s	59 MB	84.1 s	53 MB
SWB3	187.3 s	76 MB	200.2 s	64 MB

Table 1: A comparison of the execution time and memory requirements of the ISIP decoder for Alphadigits and SWB.

4. EXPERIMENTS AND RESULTS

We conducted several lattice rescoring experiments using the OGI Alphadigits (AD) and SWB corpora to demonstrate the efficiency of the ISIP decoder. A commercial decoder, HVite, operating under identical constraints (same lattices and acoustic models, equal global beam width) was used as the baseline. Some results on utterances of various lengths and complexities are displayed in Table 1. While the ISIP decoder runs faster than the baseline system, its slightly higher memory usage can be attributed to not pruning unused lexical trees (under implementation), and keeping the state and model level path information alive at all times.

For an inherently large-scale task such as SWB, where the salient problems include the heavily band-limited and often noisy audio quality, extremely large vocabulary with many pronunciation and accent variants, a very fluid language model, and a high degree of coarticulation at word boundaries; the word error rate (WER) suffers in the form of numerous insertions, deletions and substitutions. It is imperative to use a decoder that can efficiently handle cross-word phonetic context.

As a performance metric of the ISIP decoder, we ran detailed evaluations on 57 utterances chosen from the 1997 Summer Workshop's development test set. We compared the performance of the ISIP decoder in terms of recognition accuracy, execution time and memory requirements under similar thresholds for pruning with HVite. The results for this task are tabulated in Table 2.

Performance	ISIP	Baseline
Insertions	4.9%	4.6%
Deletions	9.0%	8.8%
Substitutions	25.7%	26.4%
Correct words	65.3%	64.8%
Word error rate	39.6%	39.9%

Table 2: A comparison of the lattice rescoring recognition performance of the ISIP decoder for 57 SWB utterances.

While effective pruning is critical for the efficiency of an LVCSR system, most pruning criteria rely heavily on heuristics and need to be studied in detail for maximizing their utility. Figure 3 depicts the effect of the various pruning thresholds for a typical SWB lattice rescoring application using the ISIP decoder. As expected, the MAT pruning applies strict limits on the memory usage. However, the corresponding drop in execution time is not proportional. This can be attributed to the fan-out caused by surviving word-end traces, as well as the computational overhead for sorting the traces at each frame. The global and word-end beams provide a more direct control for this problem, and therefore a combination of all three techniques turns out to be most effective.

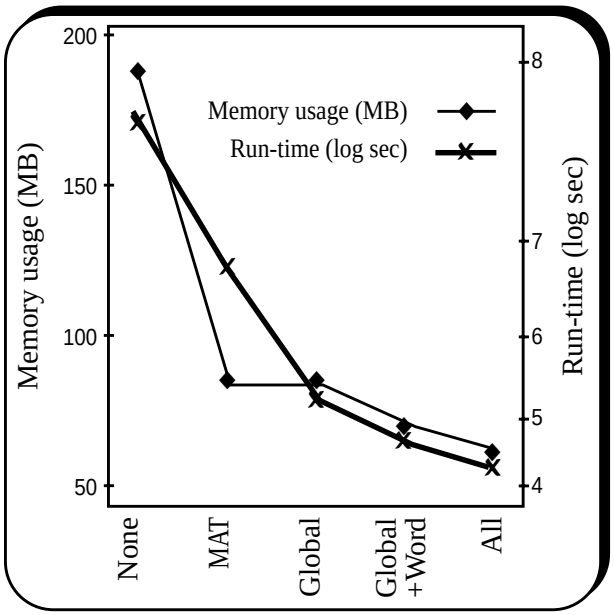

Figure 3: Effect of various pruning criteria on the performance of the ISIP decoder for a typical SWB utterance.

5. CONCLUSIONS

We have presented a new, public domain decoder that is state-of-the-art in terms of recognition performance as well as consumption of CPU and memory resources. Moreover, it has a better if not identical word error rate and it runs faster than the baseline commercial decoder. The latest release of the decoder can be freely downloaded from [5].

Future steps towards converting this decoder into a STT toolkit will include grammar-based decoding, N-best search and lattice generation, acoustic feature extraction and HMM training. We expect the new entrants to the speech research community to greatly benefit from such a free toolkit in getting started on algorithmic research.

6. REFERENCES

[1] S. J. Young, N. H. Russell and J. H. S. Thornton, "Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems", *Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR.38*, Cambridge University, 1989.

[2] H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub, "Progressive-Search Algorithms for Large-Vocabulary Speech Recognition", in *Proceedings of the DARPA Human Language Technology Workshop*, March 1993.

[3] S. Ortmanns, H. Ney and A. Eiden, "Language Model Look-ahead for Large Vocabulary Speech Recognition", in *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 2095-2098, October 1996.

[4] J. J. Odell, V. Valtchev, P. C. Woodland and S. J. Young, "A One-Pass Decoder Design for Large Vocabulary Recognition", in *Proceedings of the DARPA Human Language Technology Workshop*, pp. 405-410, March 1995.

[5] N. Deshmukh, A. Ganapathiraju, J. Hamaker and J. Picone, "Large Vocabulary Conversational Speech Recognition", *http://www.isip.msstate.edu/resources/technology/projects/1998/speech_recognition/*, Mississippi State University, 1998.