AN EFFICIENT PUBLIC DOMAIN LVCSR DECODER

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 developing core technology in speech recognition highlights the need for freely available state-of-the-art software. We have released an initial version of an LVCSR decoder that supports cross-word context-dependent phone modeling and lattice rescoring. The core search engine uses a variation of the Viterbi algorithm to efficiently manage word, phone and state-level hypotheses. The decoder employs lexical trees to handle multiple pronunciations of words, and also supports general network decoding. Preliminary evaluations on the WS'97 dev test partition of SWITCHBOARD (SWB) yielded a 46.1% WER. The decoder is also shown to be competitive in computational requirements.

1. INTRODUCTION

A large vocabulary speech-to-text (STT) system consists of three main components. First, an *acoustic subsystem* converts the speech signal into a sequence of feature vectors typically modeled using Hidden Markov Models (HMMs). Next, a *linguistic* component constrains the choice of the next word given a sequence of previously recognized words. Finally, the *decoder* finds a word sequence that maximizes the likelihood of the observed acoustic evidence by searching through a large word graph. In Bayesian statistical framework, the search problem can be summarized as

$$\begin{aligned} W_t^* &= \arg max P(W_t^i \left| \boldsymbol{O}_t \right.) \\ &= \arg max P(\boldsymbol{O}_t \left| W_t^i \right.) P(W_t^i) / P(\boldsymbol{O}_t) \end{aligned} \quad (1)$$

For a state-of-the-art LVCSR system, the path calculations for decoding involve traversing through a hierarchy of graphs (sentences, words, phones, and HMM states). Also, the number of possible word sequences becomes quite large even for a small vocabulary, especially with a trigram language model (LM) and cross-word triphones. Efficient implementation of the control structure required to perform this search within reasonable system resources is quite challenging.

As a result, good decoders are always proprietary; and the investment to develop or license such quality software is prohibitive. This ultimately discourages the rate of progress in speech research. One way to decrease the overall cost of STT research is to use the Internet as a means to pool resources and provide access to the fundamental technology. The decoder presented here is part of such an Internet-based STT toolkit currently under development at the Institute for Signal and Information Processing (ISIP).

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. System Structure

The implementation of the ISIP decoder is based on a hierarchical variation of the standard Viterbi-style time-synchronous search paradigm [1]. At each frame of the utterance being decoded, the system maintains complete history for each active path at each level in the search hierarchy via special scoring data structures (markers). Each path marker keeps its bearings in the search space hierarchy by indexing the current lattice

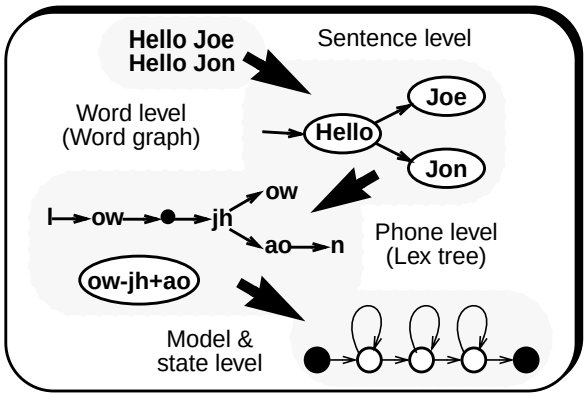

Figure 1: Illustration of the hierarchical representation of the search space in the ISIP decoder.

node, lexical tree node and the triphone model. It also maintains the path score and a back-pointer to its predecessor.

Figure 1 illustrates the hierarchical framework of the search space. For instance, at each instantiation of a triphone, a state-level path marker is projected from the previous phone-level marker and added to a state-level list of path markers. For each frame, the active states are evaluated only once. The state-level markers are compared and the best marker for each different instance of the state is projected to the next states as governed by the state transition probabilities (Viterbi decoding). The score for each state is stored locally and added to the projected path marker score. A marker exiting the model is added to the phone-level marker list and used to project the next triphone markers. Similarly, phone-level path markers at end of words are promoted to the word level and used to project paths into the subsequent words.

2.2. Lattice Compaction

During lattice rescoring, it is fairly standard practice to ignore the timing information associated with the lattice nodes and treat the lattice as a word graph constraining the search space. In this case, many arcs of the lattice indicate essentially the same word sequence and need not be decoded individually [2].

The ISIP decoder compacts the original lattice (usually by a factor of 2 to 5 for SWB lattices) into a word graph that preserves all the word hypotheses, yet merges all such duplicate arcs. This causes a significant drop in the search space complexity at minimal computational overhead.

2.3. Lexical Trees

The ISIP decoder uses lexical trees to represent the pronunciations of all words following a particular node in the lattice. A lexical tree is created only when the predecessor lattice node is reached in the decoding process, and is shared by multiple instances of that lattice node. A lexical tree no longer actively used for decoding is pruned away to save memory.

Each lexical tree node is associated with a monophone in the pronunciation of the words (see Figure 1). The node also contains the maximum LM score of all the words covered by that node. This score is used for efficient LM look-ahead [3] and appended to the path score temporarily for the sake of pruning comparisons. Once a terminal node is reached, the identity of the word is unique and the actual word LM score is added to the path score.

2.4. Dynamic Triphone Generation

Triphones are generated dynamically by traversing the lexical tree nodes at each step as illustrated in Figure 1. Cross-word triphones are created by growing (if necessary) the lexical tree corresponding to the lattice node containing the currently evaluated word. This reduces the required tree size and facilitates creation of triphones on an as needed basis.

2.5. Path Merging

If all paths in the search space are allowed to grow independently, the computational load on the decoder increases exponentially with time. By sharing the evaluation of similar parts of different hypotheses the decoder can prevent the computational overload. Hypotheses with the same acoustic and linguistic context (as determined by the position in the lattice and lexical tree hierarchy) have identical futures, and therefore can be merged into one. Here only the highest-scoring path marker is propagated for each triphone instance at such points (such as word ends).

2.6. Pruning

Two heuristic pruning techniques are employed to prevent evaluation of low-scoring hypotheses.

Beam pruning: The ISIP decoder allows the user to set a separate beam at each level in the search

hierarchy. The beam width at each level is determined empirically, and the beam threshold is computed with respect to the best scoring path marker at that level. For instance, if at a frame t the best path scores at the state, phone and word levels are respectively given by

$$\begin{aligned} q_{max}(s,t) &= max\{q(s,t)\} \ldots \forall s \\ q_{max}(p,t) &= max\{q(p,t)\} \ldots \forall p \\ q_{max}(w,t) &= max\{q(w,t)\} \ldots \forall w \end{aligned} \quad ; \quad (2)$$

then for beam widths of $b(s), b(p)$ and $b(w)$, the decoder prunes all hypotheses which satisfy

$$\begin{aligned} q(s,t) &< q_{max}(s,t) + b(s) \\ q(p,t) &< q_{max}(p,t) + b(p) \\ q(w,t) &< q_{max}(w,t) + b(w) \end{aligned} \quad . \quad (3)$$

Maximum Active Phone Model Instance (MAPMI) pruning: By setting an upper limit on the number of active triphone instances 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 *mapmi_limit*, then only the best *mapmi_limit* hypotheses are allowed to continue while the rest are pruned off.

3. SOFTWARE 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, thereby keeping the memory load on the system fairly constant. The software structure representing the hierarchical framework of the search space is extensible to higher levels such as sentences. Also, hooks are provided to apply various kinds of acoustic distributions and introducing newer modules and modalities (such as grammar decoding).

The decoder also functions in a demonstration mode interfaced with a Tcl-Tk based graphical interface (Figure 2). It provides a frame-by-frame display of the top word hypotheses, cross-word triphones and statistics on the path marker usage at different levels, serving as a valuable debugging and educational tool.

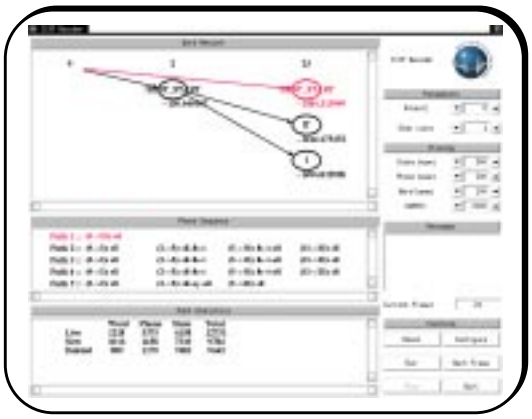

Figure 2: A screenshot of the graphical interface to the ISIP decoder demonstration mode.

4. EVALUATIONS

To gauge the performance of the ISIP decoder we ran a detailed evaluation on the WS'97 dev test — a subset of the SWB corpus. This evaluation set spans 1.56 hours of speech (2427 utterances, of durations varying from 0.5s to about 15s). The acoustic models were estimated using HTK on 60 hours of training data, and are simple three-state left-to-right HMMs. Each state is represented by a 12-mixture Gaussian distribution with diagonal covariances. The lattices were generated using HTK with around 10% inherent WER. The HTK decoder HVite was used as baseline.

4.1. Evaluation Results

A comparison of the ISIP decoder with the best performance (with optimal pruning thresholds of 200, 150, 150, 2000 respectively for state, phone and word-level beams and MAPMI, requires 72MB at most), as well as the fastest time (at 200, 50, 100, 500 pruning, 62MB) with the baseline system is shown in Table 1. The best WER of 46.1% is slightly better than the baseline. Also, reducing the run time from 30xRT to 10xRT raises the WER only by 31%.

4.2. Effect of Pruning Strategies

The choice for the individual beam widths directly affects the search space both in terms of recognition accuracy as well as CPU and memory requirements. We studied the effect of various pruning strategies on the decoder with a smaller test set of 300 utterances (see Figure 3). The decoder WER remains fairly constant till about 10 xRT on this set.

Error Type	ISIP Best (30xRT)		ISIP Fastest (10xRT)		HVite	
	WER	SER	WER	SER	WER	SER
Sub	31.4	66.2	38.7	71.7	31.1	66.8
Del	10.9	40.6	16.8	43.8	11.7	41.1
Ins	3.8	17.5	4.8	21.5	3.6	18.0
Total	**46.1**	70.1	**60.4**	74.7	**46.4**	70.9

Table 1: The recognition performance of the ISIP decoder on 2427 SWB utterances (the WS'97 dev test data). All numbers indicate percent errors.

A combination of all pruning techniques was found to be the most effective (see Figure 4). The MAPMI pruning applies strict limits on the memory usage. However, this does not result in a proportional drop in execution time due to the fan-out caused by the surviving word-end markers, as well as the computational overhead. Beam pruning provides a more direct control at each level (e.g. word-level pruning with a tighter beam to curb the LM fan-out).

5. CONCLUSIONS

We have introduced a new, public domain state-of-the-art decoder for LVCSR that is competitive in terms of both recognition performance as well as CPU and memory consumption. The best WER on the WS'97 dev-test set was 46.1%, better than the baseline decoder. We also studied in detail the effect of various pruning strategies on performance, and found that a combination of all four heuristics works best for this task. Moreover, a speed-up from 30xRT to 10xRT resulted in a 31% increase in WER.

The decoder source code can be 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.

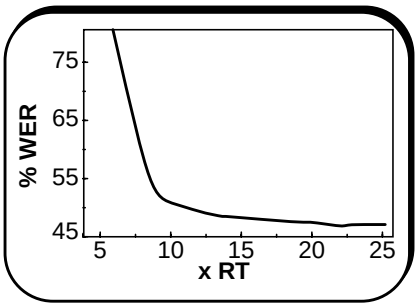

Figure 3: As the computation time decreases with tighter pruning, the WER increases.

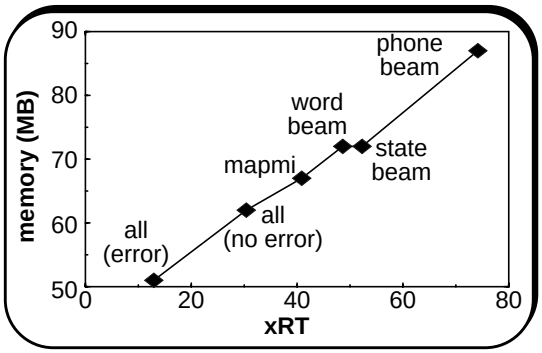

Figure 4: Effect of pruning on memory and CPU usage in the ISIP decoder on a sample test utterance. The start-up memory needed for loading models and lexicon is 48MB.

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.