MODELING AND EFFICIENT DECODING OF LARGE VOCABULARY CONVERSATIONAL SPEECH

Michael Finke, Jürgen Fritsch, Detlef Koll and Alex Waibel
(finkem,fritsch+,koll,ahw)@cs.cmu.edu
Interactive Systems Inc. Pittsburgh (USA)

ABSTRACT

Capturing the large variability of conversational speech in the framework of purely phone based speech recognizers is virtually impossible. It has been shown earlier that suprasegmental features such as speaking rate, duration and syllabic, syntactic and semantic structure are important predictors of pronunciation variation. In order to allow for a tighter coupling of these predictors of pronunciation, duration and acoustic modeling a new recognition toolkit has been developed. The phonetic transcription of speech has been generalized to an attribute based representation, thus enabling the integration of suprasegmental, non-phonetic features. A pronunciation model is trained to augment the attribute transcription to mark possible pronunciation effects which are then taken into account by the acoustic model induction algorithm. A finite state machine single-prefix-tree, one-pass, time-synchronous decoder is presented that efficiently decodes highly spontaneous speech within this new representational framework.

1. INTRODUCTION

Most speech recognition systems rely on dictionaries that contain few alternative pronunciations for most words. In natural speech, however, words seldom adhere to their citation forms. The failure of ASR systems to capture this important source of variability is potentially a significant source for recognition errors, particularly in spontaneous, conversational speech. The availability of phonetically transcribed corpora led to work on automatic inference of pronunciation variation. Unfortunately, increasing the number of variants per dictionary entry based on a pronunciation model means increasing the confusability between dictionary entries, and thus often leads to an actual performance decrease. In [3] we have introduced speaking mode as means to reduce confusability by probabilistically weighting alternative pronunciations depending on the speaking style. Pronunciation modeling and acoustic modeling were introduced as being dependent on a wider range of observables ranging from speaking rate and durations to syllabic, syntactic and semantic structure. All of these contributing factors were subsumed in the notion of speaking mode [7].

In this paper we present a new speech recognition toolkit that was specifically designed to allow for this more flexible modeling of conversational speech. The notion of context is broadened from its purely phonetic definition to a context that incorporates all sorts of features and predictors: dialect, gender, word or syllable position, durations, speaking rate, fundamental frequencies, HMM state etc.. This affects all levels of modeling within the recognition engine, from the way words are represented in the dictionary, through pronunciation modeling, duration modeling to acoustic modeling. In the second part of the paper we will introduce strategies to efficiently decode conversational speech within the mode dependent modeling framework. A one-pass, time synchronous decoder is described and evaluated on Switchboard, a human-to-human telephone corpus.

2. MODELING CONVERSATIONAL SPEECH

Pronunciation differences represent one important source of variability in spontaneous, conversational speech that is not well accounted for by current recognition systems. The speaking mode dependent modeling framework proposed earlier [7, 3] has shown significant gains in terms of word accuracy through a more detailed modeling of the surface form of the pronunciation.

2.1. Pitfalls of Phonetic Represention

Just as the phonetic representation of careful speech is a schematization of articulatory and acoustic events, a phonetic transcription of sloppy speech must be a gross simplification: pronunciation models that implement purely phonological mappings generate phonetic transcriptions which are underspecified in terms of durational and spectral properties. Thus, reduced variants as predicted by a pronunciation model end up being expected to be phonetically homophonous (e.g. the fast variant of "support" being pronounced as /s/p/o/r/t/ phonetically homophonous with "sport"). But, for such homophony to be created, not only would the unstressed vowels have to be deleted, but the durations of the remaining phones would have to take exactly the same values that they have in words not derived via fast speech vowel reduction. Similarily, fast speech intervocalic voicing in a word like "faces" cannot be precisely represented as /f/ey/z/ih/z/ phonetically homophonous with "phases", unless both the voice value of the fricative as well as the durational relationship between the stressed vowel and the fricative have changed.

2.2. From Phones to Attribute Instances

Instead of starting from a purely phonetic representation where a word is transcribed as a sequence of phones our recognition toolkit is based on attributes. Starting point is a set of attributes each of which can be either binary, discrete (i.e. multi-valued), or continuous valued. The basic set of attributes used to build a speech recognizer are articulatory features (e.g. vowel, high, nasal, reduced), stress, word position (e.g. word begin/end, syllable boundary), word class (e.g. pause, function word) and HMM state (e.g. begin/middle/end state). A word w is transcribed as a sequence of instances $(\iota_0\iota_1\ldots\iota_k)$ which are bundles of instantiated attributes (i.e. attribute-value pairs). The filled pause "um" for example is transcribed by a single instance ι consisting of truth values for the following binary attributes (pause, nasal, voiced, labial...).

2.3. Pronunciation Modeling

The instance based representation allows for a more careful modeling of pronunciation effects as observed in sloppy speech. Instead of predicting the expected phonetic surface form based on a purely phonetic context we augment the canonical instance-based transcription probabilistically. The pronunciation model predicts instances for a set of attributes: reduced, deleted, nasalized, shifted, duration, speaking rate etc.. That means that instead of mapping from one phone sequence to another as described in 2.1. the pronunciation model is trained to predict pronunciation effects/phenomena:

$$p({\iota'}_k | \ldots \iota_{k-1}[\iota_k]\iota_{k+1}\ldots)$$

The pronunciation variants are derived by augmenting the initial transcription by the predicted instances

$$\iota_0\iota_1\ldots\iota_k \quad\mapsto\quad (\iota_0 \oplus {\iota'}_0)(\iota_1 \oplus {\iota'}_1)\ldots(\iota_k \oplus {\iota'}_k)$$

and they are weighted by the probability

$$p({\iota'}_0{\iota'}_1\ldots{\iota'}_k) \quad=\quad \frac{1}{Z}\prod_{\kappa=0}^{k} p({\iota'}_\kappa | \ldots \iota_{\kappa-1}[\iota_\kappa]\iota_{\kappa+1}\ldots)$$

where Z is a normalizing constant.

Prediciting pronunciation variation by means of augmenting the phonetic transcription by expected pronunciation effects as described above avoids possibly homophonous representation of variants [3]. The original transcription is preserved and it is left to the duration and acoustic model building process to exploit the augmented annotation.

2.4. Acoustic Modeling

Decision trees are grown to induce a set of context dependent duration and acoustic models. The induction algorithm allows for questions with respect to all attributes defined in the transcription. Thus, starting from the augmented transcription context dependent modeling means that the acoustic models derived depend on the phonetic context as well as the pronunciation effects and all speaking mode related attributes. This leads to a much tighter coupling of pronunciation modeling and acoustic modeling because model induction takes the pronunciation predictors into account as well as acoustic evidence.

3. FINITE STATE MACHINE DECODER

The primary goal when designing the LVCSR decoder was to build a toolkit as flexible as possible to do research on conversational speech as described above and at the same time allow for efficient decoding runs. For the sake of coherence of training, testing and rescoring results the same decoder was supposed to take care of finite state grammar decoding and forced alignment of training transcripts, large vocabulary statistical grammar decoding and lattice rescoring. In the following we present a single-prefix-tree time-synchronous one-pass decoder which is based on abstract finite state machines to represent the underlying grammar to be recognized.

3.1. Single-Prefix-Tree One-Pass Decoder

To achieve reasonable efficiency in a one-pass decoder the dictionary has to be represented by a pronunciation prefix tree [5]. The two problems due to this representation are, first, if the tree is reentrant then only the single best history is considered at word transitions at each time t and second, the application of the grammar score is delayed since the identity of the word is only known at the leaves of the tree.

Our approach of dealing with the first problem is to have a priority heap to represent alternative linguistic theories in each node of the prefix tree as described in [1]. The heap maintains all contexts whose probabilities are within a certain threshold thus avoiding following the single best local history only. The threshold and the heap policy have the benefit of allowing us to employ different more or less aggressive search techniques by effectively controlling hypothesis merging. In contrast to the tree copying process as employed by other recognizers the heap approach is more dynamic and scalable.

3.2. Finite State Language Model Interface

The language model is presented to the decoder by means of an abstract finite state machine representation. The exact nature of the underlying grammar remains transparent to the recognizer. The only means to interact with a respective language model is through the following set of functions. Let FSM be a finite state machine based language model:

FSM.initial() Returns the initial state of the FSM.

FSM.arcs(state) Returns all arcs departing from a given state. An arc consists of the input label (recognized word), the output label, the cost and the next state. Finite state machines are allowed to be nondeterministic, i.e. there are possibly two arcs with the same input label.

FSM.cost(state) Returns the exit cost for a given state to signal whether a state is a final state or not.

This abstraction of the language model interface makes merging of linguistic theories a straight forward and well defined task to the decoder: two theories fall into the same congruence class of histories and thus can be merged if the state indices match. The finite state machine is supposed to return which theories can be merged. The advantage of this division of labor is that the decoder can without any additional implementation effort decode grammars of any order.

In order to deal with filler words, i.e. words that are not modeled by a particular FSM grammar (these are typically pauses such as silence and noises), the decoder virtually adds a self loop with a given cost term to each grammar state. As a result any number of filler words can be accepted/recognized at each state of the finite state machine.

The toolkit provides a set of different instantiations of the finite state machine interfaces which are used in different contexts of training, testing or rescoring a recognizer:

- Finite State Grammar Decoding: The most immediate application of the FSM interface idea is to define a finite state grammar explicitly. Besides its use in command-and-control applications we employ this feature in the course of training the recognizer. In [2] we have shown that when dealing with unreliable transcripts of the training data a significant gain in word accuracy can be achieved by training from probabilistic transcription graphs instead of the raw transcripts. The toolkit allows for decoding of right recursive rule grammars by simulating an underlying heap to deal with recursion. The transcription graphs of the Flexible Transcription Alignment (FTA) paradigm are expressed in the decoder in terms of a probabilistic rule grammar. Thus, forced alignment of the training data is basically done through decoding these utterance grammars.

- N-gram Decoding: Statistical n-gram language models are not explicitly represented as a finite state machine. Instead a finite state machine wrapper is built around n-gram models. The state index codes the history such that FSM.arcs(state) can retrieve all the language model scores required from the underlying n-gram tables. This implies that the FSM is not minimized and the state space is the vocabulary to the power of the order of the n-gram model.
- Lattice Rescoring: Lattices are finite state machines, too. So rescoring a word graph using a different set of acoustic models and a different language model is feasible by means of decoding along lattices and by on-the-fly composition of finite state machines.

3.3. Finite State Machine Lookahead

The incorporation of the grammar probabilities into the search process should be done as early as possible so that tighter pruning thresholds can be used for decoding [6]. Within the finite state machine abstraction the lookahead techniques of [6] can be generalized to any kind of FSM based language model. For each state the decoder needs to derive - on demand - a cost tree which reports for every node of the prefix tree what the best language model score for all words with a given prefix is going to be. For a trigram based FSM the lookahead tree will consequently be a trigram lookahead, for fourgrams a fourgram lookahead and for finite state grammars the lookahead will be projection of all words allowed at a certain grammar state. In order to compute finite state machine lookahead trees efficiently on demand several techniques needed to be combined:

- Lookahead trees once computed are saved in an aging cache to avoid recomputing the tree for subsequent frames.
- To reduce the size of the cache and the number of steps to compute the tree we precompute from the prefix tree a new data structure: the cost tree. The cost tree represents the cost structure in a condensed way and turns the rather expensive recursive procedure of finding the best score in the tree into an iterative algorithm.
- Each heap element or hypothesis or tree copy has the current FSM lookahead score attached. When the hypothesis is expanded to the next node and the respective lookahead tree has been removed from the cache in the meantime the tree will not be recomputed. Instead we propagate the lookahead probability of the prefix ("lazy cache" evaluation).

3.4. Crossword Modeling

The acoustic models we build are polyphonic within-word models but triphone models across word boundaries. To incorporate crossword modeling in a single-prefix-tree decoder we have to deal with context dependent root and leaf nodes. Instead of having context dependent copies of the prefix tree each root node is represented as a set of models, one for each possible phonetic context. The hypotheses of these models are merged at the transition to the within word units (fan-in). As compact means of representing the fan-in of root nodes and the fan-out of leaf nodes we introduced the notion of a multiplexer. A multiplexer is a dual map that maps instances ι to the index of a unique hidden markov model which is supposed to be the model to be used in the context of ι:

$$\begin{aligned} \mathrm{mpx}(\iota) &\;:\; \iota \mapsto i \in \{0, 1, \ldots N_{\mathrm{mpx}}\} \\ \mathrm{mpx}[i] &\;:\; i \mapsto m \in \{m_0, m_1, \ldots m_{N_{\mathrm{mpx}}}\} \end{aligned}$$

where $m_0, m_1 \ldots m_{N_{\mathrm{mpx}}}$ are unique models. The set of multiplexer models can be precomputed based on the acoustic modeling decision tree and the dictionary of the recognizer.

For modeling converstational speech the concept of multiplexers became particularly important since the augmented attribute representation of words leads inevitably to an explosion of the number of different crossword contexts. Since multiplexers map to unique model indices they basically implement a compression of the fan-in/out and a scheme to address the context dependent model by the context instance ι.

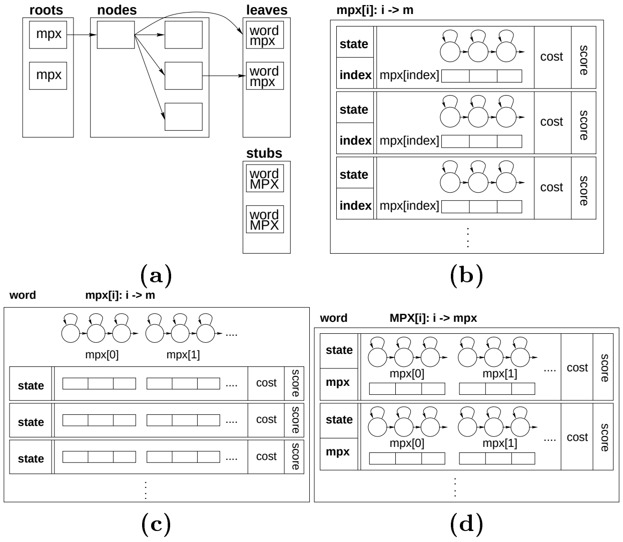

Figure 1. (a) Prefix search tree consisting of roots, nodes, leaves and single phone word nodes [stubs]. The heap structure of a root node (b), a leaf node (c), and a stub (d): state=finite state machine grammar state; mpx=multiplexer; MPX=multiplexer of multiplexers; cost=FSM lookahead score; score=total best score of hypothesis (acoustic plus expected FSM cost).

The resulting prefix tree used by the decoder consists of the following types of nodes

- Root Node: A root node represents the first attribute instance of words in terms of the respective multiplexer. The heap policy is to merge only those hypotheses that have the same history or linguistic theory and whose final instances ι_a and ι_b map to the same context dependent word initial model, i.e. $\mathrm{mpx}(\iota_a) = \mathrm{mpx}(\iota_b)$. This means that the heap is used to keep track of different contexts, the FSM state (representing the linguistic context) as well as acoustic contexts.

- Internal Node: In word internal nodes only those hypotheses are collapsed that are found to be in the same finite state machine state.
- Word/Leaf Node: For every word there is a leaf node. A multiplexer describes the fan-out. Each heap element represents the complete fan-out for a given grammar state.
- Single-Phone/Instance Node: Words consisting of one phone only are represented by a multiplexer of multiplexers. Depending on the left context of the word this multiplexer returns a multiplexer representing the right-context dependent fan-out of this word. The heap policy is the same as for root nodes, and each heap element represents the complete fan-out as for leaf nodes.

Figure 1 shows the different heap architectures for each of the prefix tree's node types.

3.5. Pruning

In addition to the acoustic and the word end beam for pruning the acoustics we use two heap related controls: the maximum number of heap elements can be bounded and there is a beam to prune hypotheses within a heap against each other. The number of finite state machine states expanded at each time t can be constrained as well (topN threshold).

3.6. Dynamic Frame Skipping (DFS)

Acoustic model evaluation is sped up by means of gaussian selection through Bucket Box Intersection [4] and by dynamic frame skipping (DFS): The underlying idea here is to reevaluate acoustic models only provided the acoustic vector changed significantly from time t to time $t+1$. A threshold on the euclidean distance is defined to trigger reevaluation of the acoustics. To avoid skipping too many consecutive frames we allow only for one skip at a time, i.e. after skipping one frame the next one must be evaluated.

4. SUMMARY

We have presented a new recognition toolkit designed specifically for dealing with large vocabulary spontaneous speech. Based on a generalized representation of the phonetic transcription we recast the mode dependent pronunciation modeling approach introduced in [3]. The architecture of a new finite state machine based one-pass decoder, the heap organization, lookahead infrastructure and crossword handling were described and motivated.

To assess the performance of the decoder under tight realtime constraints we started from a Switchboard recognizer trained on human-to-human telephone speech. The acoustic frontend computes 42 dimensional feature vectors consisting of 13 mel-frequency cepstral coefficients plus log power and their first and second derivatives. Cepstral mean and variance normalization as well as vocal tract length normalization are used to compensate for channel and speaker variation. The recognizer consists of 8000 pentaphonic Gaussian mixture models. A 15k words recognition vocabulary and approximately 30k dictionary variants generated by a mode dependent pronunciation model are used for decoding. Without MLLR adaptation and decoded with a Switchboard trigram language model trained on 3.5 million words the base performance at 100xRT is 37% word error rate (run-on, one-pass recognition on NIST Eval'96). Groups participating in recent NIST evaluations reported decoding times in the order of 300 realtime factors [which includes multiple adaptation passes]. Table 1 shows the first word accuracy results of our Switchboard recognizer at around ten times realtime.

Condition	RT	WER
Baseline	100	37%
Tight beams, topN=10	12	43.8%
Tight beams, topN=10, DFS	7	45.6%
Tight beams, topN=10, DFS, BBI	5	49.8%

Table 1. Tight pruning in the context of highly confusable Switchboard speech (topN=10 means that only 10 finite state machine states were expanded per frame; DFS=Dynamic Frame Skipping; BBI=Bucket Box Intersection).

REFERENCES

[1] F. Alleva, X. Huang, and M.Hwang. Improvements of the Pronunciation Prefix Tree Search Organization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, Georgia USA, May 1996.

[2] M. Finke and A. Waibel. Flexible Transcription Alignment. In *1997 IEEE Workshop on Speech Recognition and Understanding*, Santa Barbara, California, December 1997.

[3] M. Finke and A. Waibel. Speaking Mode Dependent Pronunciation Modeling in Large Vocabulary Conversational Speech Recognition. In *Proceedings of Eurospeech*, volume 5, pages 2379–2382, September 1997.

[4] J. Fritsch and I. Rogina. The Bucket Box Intersection (BBI) Algorithm for Fast Approximative Evaluation of Diagonal Mixture Gaussians. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, Georgia USA, May 1996.

[5] H. Ney, R. Haeb-Umbach, B.H. Tran, and M. Oerder. Improvements in Beam Search for 10000 Word Continuous Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1992.

[6] S. Ortmanns, A. Eiden, H. Ney, and N. Coenen. Look-Ahead Techniques for Fast Beam Search. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1783–1786, Munich, 1997. IEEE.

[7] M. Ostendorf, B. Byrne, M. Bacchiani, M. Finke, A. Gunawardana, K. Ross, S. Roweis, E. Shriberg, D. Talkin, A.Waibel, B. Wheatley, and T. Zeppenfeld. Systematic Variations in Pronunciation via a Language-Dependent Hidden Speaking Mode. In *International Conference on Spoken Language Processing*, Philadelphia, USA, 1996.