THE BBN SINGLE-PHONETIC-TREE FAST-MATCH ALGORITHM

Long Nguyen and Richard Schwartz

BBN Technologies, GTE Internetworking
Cambridge, MA 02138, USA
ln@bbn.com

ABSTRACT

In this paper we present a very fast and accurate fast-match algorithm which, when followed by a regular beam search restricted within only the subset of words selected by the fast-match, can speed up the recognition process by at least two orders of magnitude in comparison to a typical single-pass speech recognizer utilizing the Viterbi (or Beam) search algorithm. This is a novel fast-match algorithm that has two important properties: high-accuracy recognition and run-time proportional to only the cube root of the vocabulary size.

1. INTRODUCTION

In January 1993, at a DARPA workshop held at MIT, BBN demonstrated for the first time ever a real-time, 20K-word, speaker-independent, continuous speech recognition system, implemented in software on an off-the-shelf workstation. One part of the algorithm was published soon thereafter [1]. However, the fast-match part of the algorithm, which has recently received a US patent [2], has not been described until now. While a number of fast-match algorithms have been published, the BBN algorithm continues to have novel features that have not appeared in the literature. The fast-match algorithm has two important properties: high-accuracy recognition and run-time proportional to only the cube root of the vocabulary size.

In this algorithm, the vocabulary is organized as a phonetic tree similar to Ney's [3]. However, in contrast to prior approaches in which several copies of the trees are needed in order to use a word bigram language model, the innovation in this algorithm allows us to use a word bigram language model with just a single phonetic tree. In the remainder of the paper, we will present in detail how we construct such a phonetic tree and how to estimate the acoustic and language models. Then we will explain the search algorithm and report our finding about the computational requirements.

2. PHONETIC TREE

Assume a vocabulary that consists of the following three words, '*abc*', '*abcd*', and '*ade*' whose phonetic pronunciations are **a-b-c**, **a-b-c-d**, and **a-d-e** respectively. The phonetic tree for this vocabulary can be constructed as illustrated in Figure 1. The annotations next to the nodes are the *composite* triphones of the phonemes associated with the nodes. For example, **&[a]b,d** denotes a *composite* triphone for phoneme **'a'** whose left context is the boundary with the special symbol **'&'**, and right context is phonemes **'b'** or **'d'**. That is, the special triphone **&[a]b,d** is a *composition* of two normal triphones **&[a]b** and **&[a]d**. All other nodes can be intepreted in the same manner; for example, **b[c]d** is a composite triphone for phoneme **c** whose left context is phoneme **b** and right context **d**.

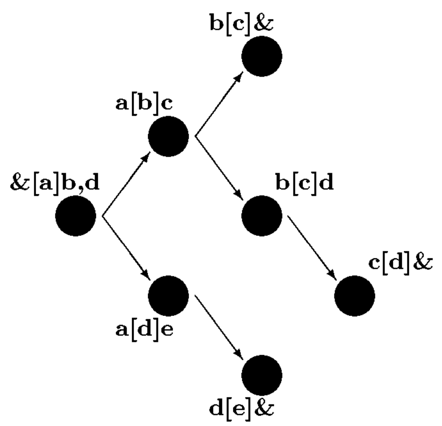

Figure 1: The phonetic tree for a hypothetical lexicon of three words *'abc', 'abcd',* and *'ade'*.

There are probably some unique characteristics of this type of phonetic tree in comparison to other types of lexical trees studied before, such as that of Ney's [3]. First, the phonemes are associated with the nodes rather than the arcs of the tree. Second, the last phoneme node of each word is kept unique, even if the word is a substring of another word. Third, each node in the tree is associated with a *set id* representing the set of words which share this node. The last two characteristics of this type of phonetic tree make it possible to use a word bigram language model during the search without tree copying.

3. ACOUSTIC MODELS

With the phonetic tree constructed as in Figure 1 where each node represents a (possibly shared) triphone, the acoustic model for the composite triphone associated with that node can be approximated as a weighted average of the correspondent normal triphones. For a Tied Mixture (TM) or Phonetically-Tied Mixture (PTM) or State-Clustered Tied-Mixture (STM) model [4], the correspondent normal triphones share the same codebook (i.e. a Gaussian mixture) whereas their mixture weights are separate. Then the composite triphone would use that same codebook and its mixture weights are the weighted average of the mixture weights of the correspondent normal triphones. For example, the mixture weight $v_j^{\&[\mathbf{a}]\mathbf{b},\mathbf{d}}$ for component j of the Gaussian mixture for composite triphone **&[a]b,d** is calculated as:

$$v_j^{\&[\mathbf{a}]\mathbf{b},\mathbf{d}} = \frac{v_j^{\&[\mathbf{a}]\mathbf{b}} * c^{\&[\mathbf{a}]\mathbf{b}} + v_j^{\&[\mathbf{a}]\mathbf{d}} * c^{\&[\mathbf{a}]\mathbf{d}}}{c^{\&[\mathbf{a}]\mathbf{b}} + c^{\&[\mathbf{a}]\mathbf{d}}}$$

where v_j^x stands for the mixture weight for component j of the normal triphone x and c^x stands for the EM (training) count of the normal triphone x.

4. LANGUAGE MODELS

In the same manner as done for the acoustic models, we can approximate a bigram language model for these nodes as well. In contrast to previous approaches where the language bigram probability is applied either at the first phoneme or the last phoneme node with some form of tree copying, our algorithm allows us to apply the language probability *cumulatively* over all composite triphone nodes of the words in the single tree. Assume the same phonetic tree as in Figure 1, and some word w has just ended, we want to apply the probability of going into node **&[a]b,d**. Since all three words, *'abc'*, *'abcd'*, and *'ade'*, share this node, the probability of going into node **&[a]b,d** given the preceding word w would be

$$Pr(\&[\mathbf{a}]\mathbf{b},\mathbf{d}|w) = Pr(abc|w) + Pr(abcd|w) + Pr(ade|w).$$

Similarly,

$$Pr(\mathbf{a}[\mathbf{b}]\mathbf{c}|w) = Pr(abc|w) + Pr(abcd|w),$$

$$Pr(\mathbf{b}[\mathbf{c}]\&|w) = Pr(abc|w),$$

$$Pr(\mathbf{b}[\mathbf{c}]\mathbf{d}|w) = Pr(\mathbf{c}[\mathbf{d}]\&|w) = Pr(abcd|w),$$

and

$$Pr(\mathbf{a}[\mathbf{d}]\mathbf{e}|w) = Pr(\mathbf{d}[\mathbf{e}]\&|w) = Pr(ade|w).$$

We call this language model a *composite set bigram model* since it is the collection of the conditional probability of a set of words that share a composite triphone given a preceding word. Note that, since the last phonemes of the words are not shared, the sets associated with the leaves are singletons (i.e. sets which consist of a single member). Consequently, the conditional probability of the set at the last phoneme of a word given a preceding word is just the usual word bigram probability.

The lower-order composite set ngrams (i.e. the set unigrams) can alse be approximated in the same manner.

To say it another way, the composite set bigram language model used in this fast-match is a different representation of the usual word bigram language model with some additions. First, the usual $Pr(w_i|w_j)$ now becomes $Pr(\{w_i\}|w_j)$, where $\{w_i\}$ is the singleton set that consists of only w_i. $Pr(w_i)$ becomes $Pr(\{w_i\})$. For some set s_i which includes more than one member, $Pr(s_i|w_j) = \sum_{\forall w_k \in s_i} Pr(w_k|w_j)$. And $Pr(s_i) = \sum_{\forall w_k \in s_i} Pr(w_k)$.

5. THE SEARCH ALGORITHM

The search algorithm is similar to the time-synchronous beam search [5] with a small addition to use the composite set bigrams. Again, assume the same phonetic tree as before, at some time t, some k words end. Let α_i^t be the partial path score from the beginning of the sentence up to word w_i at time t, node **&[a]b,d** will be activated with the product score

$$s = \arg\max_{1 \le i \le k} \{\alpha_i^t * Pr(\&[\mathbf{a}]\mathbf{b},\mathbf{d}|w_i)\}. \qquad (1)$$

That is, we search over the k ending words for the best word to go into node **&[a]b,d**. The value of s and the time t are then associated and carried along with node **&[a]b,d** during its duration. At some t_1 frames later, with an exit score s', **&[a]b,d** will activate **a[b]c** and **a[d]e** with the products

$$u = \arg\max_{1 \le i \le k} \{\alpha_i^t * Pr(\mathbf{a}[\mathbf{b}]\mathbf{c}|w_i)\} * \frac{s'}{s}$$

and

$$v = \arg\max_{1 \le i \le k} \{\alpha_i^t * Pr(\mathbf{a}[\mathbf{d}]\mathbf{e}|w_i)\} * \frac{s'}{s}$$

respectively. Note that, we still search over the same k ending words at time t. Both **a[b]c** and **a[d]e** carry along with them the time t, and the values u and v respectively. Note that the division s'/s in effect takes out the temporary composite set bigram $Pr(\&[\mathbf{a}]\mathbf{b},\mathbf{d}|w_i)$ used in the preceding node. This is the case since s' is the product of s and the acoustic score for node **&[a]b,d** from time t to time $t + t_1$.

Then after some t_2 frames later, assume that node **a[b]c** ends with an exit score u'. In turn, **a[b]c** will activate **b[c]&** and **b[c]d** with the products

$$p = \arg\max_{1 \le i \le k} \{\alpha_i^t * Pr(\mathbf{b}[\mathbf{c}]\&|w_i)\} * \frac{u'}{u}$$

and

$$q = \arg\max_{1 \le i \le k} \{\alpha_i^t * Pr(\mathbf{b}[\mathbf{c}]\mathbf{d}|w_i)\} * \frac{u'}{u}$$

respectively.

Recall that by the design of the phonetic tree, the composite set associated with the node representing the last phoneme of the word is a singleton set. So, for node **b[c]&**, $Pr(\mathbf{b[c]\&}|w_i) = Pr(\{abc\}|w_i) = Pr(abc|w_i)$. Consequently, the search algorithm really uses a true word bigram language model when it reaches the last phoneme of the word. All other set bigrams used for the interior nodes could be considered as partial or temporary language model scores. The gradual amortization of the language model score makes pruning much more efficient and robust.

Eventually, node **b[c]&** will end, say at t_3 frames later, and the search will cycle back to the propagation mentioned in Equation 1 for the root node of the phonetic tree with a new value $s_{abc}^{t+t_1+t_2+t_3}$. As reflected in Equation 1, the word bigram $Pr(\mathbf{b[c]\&}|w_i)$ is *not* taken out (as those composite set bigrams at the interior nodes are, through the division s'/s and u'/u, etc...).

In general, the propagation of theories on this phonetic tree is quite similar to that of a beam search on a linear lexicon, except for the addition of the adjustment of the composite set bigrams when approaching a phoneme node: To activate a node, we temporarily use some composite bigram probability; to leave that node, we remove that temporary bigram probability. The closer the search approaches the end of the word, it uses a more complete bigram probability.

5.1. Normalized Forward-Backward

As described in [6], the only goal of this fast-match is to keep the likely word endings and their partial scores to guide the second pass. This can be simply done by maintaining a list of words ending at each frame and their partial scores. At each time frame, we record the score of the final state of each word ending. Let Ω^t be the set of words ending at time t, and $\alpha_{w_i}^t$ be the partial path score up to word w_i at time t. Each $\alpha_{w_i}^t$ represents the probability of the speech from the beginning of the utterance up to time t given the most likely word sequence ending with word w_i times the probability of the language model for that word sequence.

As described in [7] and [8], the second backward pass is essentially the time-synchronous beam search. When some word w ends at some time t with a partial score β_w^t (β is similar to α in the forward pass but from the end of the utterance up to w), instead of activating the whole lexicon as in the linear lexicon beam search, we only activate those words $w_i \in \Omega^{t-1}$ if they satisfy the following condition:

$$\frac{\alpha_{w_i}^{t-1}}{\max \alpha^{t-1}} * \frac{\beta_w^t}{\max \beta^t} * Pr(w_i|w, w_j) > \gamma$$

where w_j is the best 'preceding' word of w, and γ is the forward-backward pruning threshold.

5.2. Admissibility

The fast-match algorithm is clearly not admissible in a strict sense. However, although the best result from the fast-match is not as accurate as the full search, we find in very large studies that it never causes increased error for the second pass.

5.3. Efficiency Issues

It is possible to make the fast-match run as fast as possible provided that it can save sufficiently good words ending at each frame for the second pass. We typically save about 100 words. The first thing that can speed up the search is to minimize k in Equation 1 (This also helps all the other arg $\max_{1\leq i\leq k}\alpha_i^t$... evaluation as well). Right after saving these k words to guide the second pass later, this list can be truncated to leave only a few high-score words. Empirically, we observed that for a 20000-word demo system, k can be 4 or 5.

Another part of the computation that takes a long time is the access to the bigram probabilities, since these are normally stored in a compact representation. To avoid this, we establish a bigram cache for a few active states (ending words). For each of these states, we have a random access array of all of the bigram probabilities.

We can also save computation by not evaluating arg $\max_{1\leq i\leq k}\{\alpha_i^t * Pr(\text{a new destination node}|w_i)\}$ when the set id of the new destination node is the same as that of the source node. Instead, we use the same result evaluated when going into the source node before. This can be detected easily by checking if there is only one out arc from the source node. This is true since, from the design of the phonetic tree, if there is only one out arc at a node, the destination node has the same set id as the source node.

6. COMPUTATION VERSUS VOCABULARY SIZE

To learn how the computation of this search strategy (fast-match followed by a trigram Forward-Backward beam search [8]) grows with vocabulary size, we measured the computation required at three different vocabulary sizes: 1500 words, 5000 words, and 20000 words. The time required, as a fraction of real time, is shown plotted against the vocabulary size in Figure 2. As can be seen, the computation increases very slowly with increased vocabulary.

To understand the behavior better, we plotted the same numbers on a log-log scale in Figure 3. Here we can see that the three points fall neatly on a straight line, leading us to the conclusion that the computation grows as a power of the vocabulary size. Solving the equation gives us the formula

$$time = 0.03V^{1/3}$$

where V is the vocabulary size.

This is very encouraging, since it means that if we can decrease the computation needed by a small factor, it would be feasible to increase the vocabulary size by a larger factor, making recognition with very large vocabularies possible.

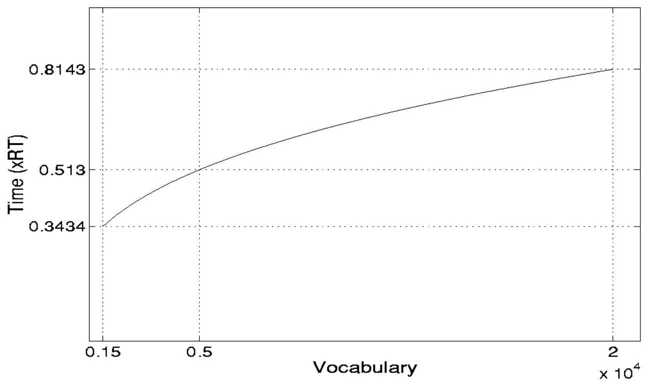

Figure 2: Run time vs. vocabulary size, linear scale, measured on an HP735 with 400 Meg RAM in 1993

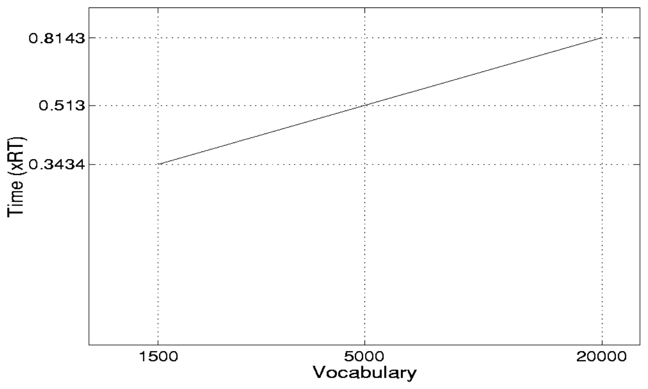

Figure 3: Run time vs. vocabulary size, log-log scale, measured on an HP735 with 400 Meg RAM in 1993

As a matter of fact, a year later in 1994, after some code optimization, this search strategy could run in less-than real time with a 40000 word vocabulary.

7. SUMMARY

We have described a novel fast-match algorithm based on a single phonetic tree. There are some unique characteristics in this proposed lexical tree which made it possible to use a word bigram language model during the search without tree copying. On this phonetic tree, all the last phonemes of the words in the lexicon always locate at the leaves of the tree. Each node of the tree is assigned a set id representing a group of words which share this node. The acoustic phoneme models associated with the nodes are the composite triphones where there could be more than one right context. In comparison to the usual triphone models, there is only a small difference for these composite triphones: the mixture weights of the composite triphones are the weighted average of the correspondent triphones. We also showed the transformation of the usual word bigram language model into a composite set bigram language model. With this composite set bigram language model, we could apply the language probabilities in a cumulative fashion at every phoneme node of the word without tree copying. The search itself is quite similar to the usual time-synchronous beam search with one addition: to activate a node, we temporarily use some composite bigram probability; to leave that node, we take out that temporary bigram probability. The fast-match can run as fast as possible provided that it can save sufficiently good words ending at each frame to guide the second pass. Finally, we demonstrated that the computation required by this algorithm grows as the cube root of the vocabulary size, which means that real-time recognition with very large vocabularies is feasible.

Acknowledgements

This work was supported in part by the Defense Advanced Research Projects Agency and monitored by Ft. Huachuca under contract No. DABT63-94-C-0063. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the Government and no official endorsement should be inferred.

8. REFERENCES

1. Nguyen, L., Schwartz, R., et al., "Search Algorithms for Software-Only Real-time Recognition", *Proc. of ARPA Human Language Technology Workshop*, Princeton, NJ, Mar. 1993, Princeton, NJ., pp. 411-414.
2. Schwartz, R., Nguyen, L., "Single Tree Method for Grammar Directed, Very Large Vocabulary Speech Recognizer", US Patent 5621859, Apr. 1997.
3. Ney, H., Haeb-Umbach, R., Tran, B.-H., Oerder, M., "Improvements in Beam Search for 10000-Word Continuous Speech Recognition", *Proc. ICASSP '92*, San Francisco, CA., Mar. 1992, pp. I.9-12.
4. Nguyen, L., Anastasakos, T., Kubala, F., LaPre, C., Makhoul, J., Schwartz, R., Yuan, N., Zavaliagkos, G., Zhao, Y., "The 1994 BBN/BYBLOS Speech Recognition System", *Proc. of ARPA Spoken Language Systems Technology Workshop*, Austin, TX, Jan. 1995, pp. 77-81.
5. Lowerre, B. T., "The Harpy Speech Recognition System", *PhD Thesis*, Carnegie-Mellon University, 1976, Pittsburgh, PA.
6. Nguyen, L., Schwartz, R., "Efficient 2-Pass N-Best Decoder", *Proc. EuroSpeech '97*, Rhodes, Greece, Sep. 1997, pp. 167-170.
7. Austin, S., Schwartz, R., Placeway, P., "The Forward-Backward Search Algorithm", *Proc. of IEEE ICASSP-91*, Toronto, Canada, May 1991, pp. 697-700.
8. Schwartz, R., Nguyen, L., Makhoul, J., "Multiple-Pass Search Strategy", *Automatic Speech and Speaker Recognition: Advanced Topics*, Kluwer Academic Publishers, Boston, 1996, pp.429-456.