A NEW VERIFICATION-BASED FAST MATCH APPROACH TO LARGE VOCABULARY CONSTINUOUS SPEECH RECOGNITION

Feng Liu *Mohamed Afify* *Hui Jiang* *Olivier Siohan*

Multimedia Communications Research Lab
Bell Laboratories, Lucent Technologies
600 Mountain Avenue, Murray Hill, NJ 07974, USA
afify,lf,hui,siohan@research.bell-labs.com

ABSTRACT

Acoustic fast match is usually used to accelerate search in large vocabulary continuous speech recognition. This paper discusses a new acoustic fast match algorithm. This proposed fast match is based on incremental evaluation of the score and the use of normalized likelihood scores. This is in contrast to more traditional fast matches where a likelihood score is used. In addition, streaming SIMD extensions (SSE) for Intel machine instructions are used for fast Gaussian calculation. Results on a 20K Japanese broadcast news task show that the proposed fast match leads to about 30% improvement in speed with a slight performance degradation.

1. INTRODUCTION

Decoding the best word sequence in large vocabulary continuous speech recognition is a computationally demanding search problem. For performing this search in reasonable time a rapid method of reducing the search space must be invoked. One of the effective techniques used for this purpose is known as acoustic fast match, or simply *fast match*.

The basic idea of a fast match is to look-ahead in time to identify, using a computationally cheap method, some phonemes with poor acoustic score and discard them before detailed evaluation[1]. While fast matches share the same basic idea they often differ in detail. Here we identify two types of fast matches:

- Global fast match (GFM), where look-ahead is done by running a Viterbi search using very simple acoustic models. Beam pruning is then applied to a node using its score and the fast match score. Examples can be found in [2, 3]. This fast match is called global because the original node score (containing global information) is used in the pruning decision.
- Local fast match (LFM), where a fixed duration look-ahead[2] and also some very simple acoustic models are used to estimate phoneme scores. These phoneme scores alone are then used as a basis for pruning (e.g. [4, 5]). Here the pruning is local because it uses only the phoneme look-ahead score.

This paper focuses on local fast match. One advantage of LFM is that the use of a fixed look-ahead interval facilitates incremental calculation of the fast match score. That is the score $S_t(\alpha)$ of a phoneme α at time t is calculated from $S_{t-1}(\alpha)$ instead of repeating the calculation at each time instant. When using hidden Markov models (HMM) an elegant recursion for this calculation can be derived [5]. Moreover, for a one state HMM or equivalently a Gaussian mixture the recursion reduces to a very simple formula involving only one subtraction and one addition.

Incremental calculation of fast match for a Gaussian mixture involves only Gaussian calculations and no time alignment calculation is involved. The size of the mixture model determines the accuracy of the fast match. A large model leads to an accurate score but with excessive computation. On the other hand a small model involves less calculation but leads to less accurate score. To address this tradeoff we use streaming SIMD extensions (SSE) for the Intel architecture instruction set to compute Gaussian likelihood [10]. These instructions considerably accelerate the calculation and hence facilitate using a larger model size for better pruning efficiency.

We consider LFM as a problem of phoneme verification. In most fast matches a likelihood based score is compared to a threshold to perform the required decision[3]. However, the use of a normalized likelihood score or likelihood ratio proved to be superior to using crude likelihoods in problems like speaker and utterance verification e.g. [6, 7]. Thus we strongly believe that using such a likelihood ratio based verification scheme will still lead to better performance of LFM. In this article we explore the use of nor-

[1] In this paper we consider fast matches for phoneme pruning. Other techniques for word pruning e.g. [1] are also referred to as fast match, but are out of the scope of this paper.

[2] A phoneme is assumed to have fixed length, e.g. its average duration, and no search for the best end point is done.

[3] In [4] the posterior is used because it is readily available from a neural network.

malized likelihood scores, using a background model, for local fast match.

To summarize, we propose a LFM for use in large vocabulary continuous speech recognition. The main novel features of this fast match compared to existing ones are the following:

- The fast match score is incrementally calculated, and the SSE instruction set is used to accelerate the computation.
- Normalized likelihood scores are used instead of the conventional likelihood scores.

The paper is organized as follows. In Section 2 we review our basic decoder structure. The proposed fast match is discussed in detail in Section 3. We finally present some experimental results on a 20K Japanese broadcast news task in Section 4 and conclude in Section 5.

2. BASIC DECODER STRUCTURE

In this section we briefly overview our two pass decoder [12]. Broadly speaking a 2-pass search consists of a very fast first pass which limits the potential search space to a set of most likely candidates which are rescored in the second pass by more detailed models. The first pass of our 2-pass decoder is based on the following:

- A single static tree search structure. This is in contrast to most tree search in speech recognition where multiple tree copies depending on the language model history are used. This single tree structure is also used by BBN [9] and NHK [8] and result in a very efficient first pass.
- Left biphone acoustic models. These models take advantage of context while still maintaining efficiency. This can be compared to averaging the triphones in the BBN system[4], and using the full triphones in the NHK recognizer.
- Bigram language models. Bigram probabilities are distributed among nodes in the tree to benefit as early as possible from the language model. This is the traditional language model lookahead used in many systems, which is efficiently implemented using caching.

Based on the above ingredients the first pass uses a classical Viterbi beam search, where at each time instant the top K candidates are propagated back into the root of the tree. The list of the best word candidates at each time instant are kept for later use in the backward pass. For additonal efficiency the forward pass makes use of word end pruning, histogram pruning, and vector quantization based fast likelihood computation. The proposed fast match is employed in this pass. At each time instant when expanding a tree node the fast match score for an arc is first consulted and used to decide whether to expand the arc or not.

The forward pass results in a set of likely words ending at each time frame. This pass is followed by a backward pass whose search space is constrained by the results of the first pass. The backward pass uses the following models:

- Within word triphone models. It was found that using cross-word triphones leads to significant increase in the search space and computation time.
- A backward trigram language model.

The search is again a Viterbi beam search employing word-end (in this case word-begin) pruning, histogram pruning, and fast likelihood computation techniques. In additon, forward-backward pruning using the scores of the first pass, and grammar spreading are used for increased efficiency.

3. FAST MATCH

Assume the score of a phoneme α at time t is denoted $S_t(\alpha)$, and that the look-ahead duration of α is d_α. We define $S_t(\alpha)$ as the log probability that α starts at t and ends at $t+d_\alpha$. Hence we write $S_t(\alpha) \equiv \log P(X_t^{t+d_\alpha}|\alpha)$, where $X_t^{t+d_\alpha}$ is the observation sequence in $[t, t+d_\alpha]$. It is interesting to write the phoneme score at time t in terms of its score at time $t-1$. If a phoneme is represented by a Gaussian mixture model, this incremental calculation can be easily written as

$$S_t(\alpha) = S_{t-1}(\alpha) + \log p(x_{t+d_\alpha}|\alpha) - \log p(x_{t-1}|\alpha) \quad (1)$$

where

$$\begin{aligned} p(x_\tau|\alpha) &= \sum_{m=1}^{M} c_m \mathcal{N}(x_\tau, \mu_m, \Sigma_m) \\ &\approx \max_m c_m \mathcal{N}(x_\tau, \mu_m, \Sigma_m) \end{aligned} \quad (2)$$

where M is the mixture size, c_m, μ_m, and Σ_m, are the mixture weight, mean, and covariance of the m^{th} Gaussian component. Also the dependence of these quantaties on the phoneme α has been dropped for convenience.

The choice of the mixture size M is of great importance. A large M gives a more accurate model and hence more reliable pruning. On the other hand a small M leads to faster computation at the expense of less accurate models. In order to reduce computation while using a large model we employ SSE instructions for Gaussian calculation. It is expected, because the fast match consists entirely of Gaussian calculations, that these instructions will significantly reduce the required computations.

Traditionally, after calculating the phoneme scores the fast match reduces to a simple test to accept α at time t if $S_t(\alpha) > \eta_\alpha$. Note that we use phone dependent thresholds. Here, we use a modified test based on normalized

[4] Essentially leading to models very similar to monophones

likelihood scores. This is motivated by the success of using normalized likelihoods in verification problems. This is achieved by replacing in equation (1) $p(x_\tau|\alpha)$ by $p_n(x_\tau|\alpha)$ given by

$$\log p_n(x_\tau|\alpha) = \log p(x_\tau|\alpha) - \log p(x_\tau|\bar{\alpha}) \qquad (3)$$

where $\bar{\alpha}$ stands for the anti-model of α. In our experiments a common background model for all phonemes was found sufficient. This is in contrast to other verification tasks where specific anti-models are used. The background model is trained on segments from all phonemes.

The fast match models are Gaussian mixture models. They are conventionally trained using a modified Kmeans algorithm. Determining the phone thresholds η_α is an important issue in designing the fast match. These are obtained as follows. Once the fast match models are trained, we evaluate the score of all segments belonging to certain phoneme α. We calculate the mean score μ_α and the standard deviation σ_α. The threshold is calculated as $\eta_\alpha = \mu_\alpha - n\sigma_\alpha$. Here n is used to tradeoff the speed and accuracy. A large n leads to a slower but more accurate search and vice versa. The choice of best n will be addressed in the experimental evaluation.

4. EXPERIMENTAL EVALUATION

The proposed algorithm is tested on a 20K vocabulary Japanese broadcast news transcription task. This work is based on a collaboration with NHK (Japan broadcasting corp.). Training and test speech data in addition to language model are provided by NHK.

The training data consists of 90 hours of speech. Context dependent hidden Markov models are built using the decision tree clustering method [11]. The feature space is 39-dimensional consisting of 12 cepstrum coefficients, energy, and their first and second order derivatives. Tags about the gender (male/female) of each segment can be included in the tree building process leading to effectively constructing gender dependent models. The resulting models have about 100K gaussians.

Trigram language models are built using Good-Turing discounting method with cutoffs =1 for the bigrams and the trigrams. The test set consists of 162 utterances from male speakers. The test set perplexity is about 34 and the out of vocabulary rate is 0.76%.

The fast match model is trained on the same data used for training the acoustic model. Fast match models are also gender dependent. The phoneme models are trained from segmented data using the K-means training algorithm. The same number of mixtures is used for every phoneme and sizes of 8,16, and 32 are used in the experiments. The background model is trained from the data of all phonemes and has the same size as the phoneme models. The look-ahead duration of each phoneme is taken as its average duration in the training data, limited to a maximum of 10 frames.

# of mix	8		16		32	
Thresh	WER	RT	WER	RT	WER	RT
1.5	11.74	0.25	11.98	0.24	11.31	0.26
2.0	7.46	0.31	6.96	0.30	7.05	0.33
2.5	5.11	0.39	5.10	0.38	5.30	0.41
3.0	4.43	0.47	4.39	0.47	4.64	0.51
3.5	4.07	0.56	4.07	0.56	4.12	0.64
4.0	3.90	0.64	3.95	0.65	4.14	0.73
∞	4.04	0.79	4.04	0.79	4.04	0.79

Table 1: Percentage word error rate (WER) and real time factor (RT) for the fast match with different mixture sizes and thresholds. Normalized scores using a background model are used here for the fast match. The row ∞ stands for the baseline result.

# of mix	8		16		32	
Thresh	WER	RT	WER	RT	WER	RT
1.5	14.24	0.52	14.05	0.52	14.24	0.61
2.0	7.64	0.62	7.56	0.62	7.24	0.72
2.5	5.20	0.70	5.18	0.71	5.14	0.84
3.0	4.48	0.78	4.28	0.79	4.55	0.95
3.5	4.12	0.83	4.14	0.86	4.12	0.64
4.0	4.07	0.87	4.09	0.91	4.09	0.99
∞	4.04	0.79	4.04	0.79	4.04	0.91

Table 2: Percentage word error rate (WER) and real time factor (RT) for the fast match with different mixture sizes and thresholds. Raw likelihood scores are used here for the fast match. The row ∞ stands for the baseline result.

This is mainly to prevent the beginning of sentence silence from having a very long look-ahead duration. The value of the threshold n is varied to study the speed/accuracy tradeoff. The computation taken by the fast match is profiled and found to be negligible comapred to the whole computational load. We explore in the following experiments the utility of introducing the normalized score instead of the raw likelihood values. The results with and without background normalization are shown in Tables 1 and 2, respectively, for different mixture size and threshold values.

From Table 1 we observe the following:

- The performance is almost insenstive to the mixture size.
- The threshold n is very important in balancing the accuracy speed tradeoff. A value of n around 3.0-3.5 accelerates the search by about 30%-40% while keeping almost the same accuracy.

Table 2 repeats the same experiments in Table 1 but without the background model normalization. From Table 1 and 2 we note the following:

	Normalized Likelihood		Likelihood	
Threshold	WER	RT	WER	RT
3.0	4.43	0.47	4.48	0.78
3.5	4.07	0.56	4.12	0.83
∞	4.04	0.79	4.04	0.79

Table 3: Percentage word error rate (WER) and real time factor (RT) with and without using the normalized likelihood scores. The size of the fast match model is 8. The row ∞ stands for the baseline result.

- Using the normalized likelihood scores significantly outperform using the raw scores for all mixture sizes and threshold values.

- Performing the fast match without normalization does not lead to any improvement over the baseline. For example, to speed up the decoding by about 30% we lose about 10% absolute recognition performance.

To further highlight the above two points Table 3 shows the word error rate and real time factor for an 8 mixture fast match model and thresholds 3, and 3.5 with and without using the normalized likelihood scores.

5. CONCLUSION

We presented an acoustic fast match for large vocabulary speech recognition. The fast match is applied in the first pass of a 2-pass search algorithm using a static tree in the first pass. When used in the search, for a 20K Japanese broadcast news task, the fast match leads to about 30%-40% speed-up in the overall search with almost no degradation in accuracy.

The fast match is based on incrementally calculating normalized likelihood scores, and the integration of SSE instruction set for fast Gaussian calculation. Incremental calculation and SSE instructions lead to rapid calculation of the fast match. The normalized scores are shown, in our experimental setting, to lead to drastic improvements in the accuracy/speed tradeoff of the fast match compared to using raw likelihood scores. This is expected because a LFM is basically a verification problem where using normalized likelihood score or likelihood ratio has proved to be superior to raw likelihood.

Acknowledgement

The authors wish to thank Dr. Akio Ando and the NHK speech recognition group for their collaboration on this project and providing the acoustic data and language model used to setup this sytem.

6. REFERENCES

[1] L. Bahl, S.V. De Gennaro, P.S. Gopalakrishnan, and R.L. Mercer, “A fast approximate acoustic match for large vocabulary speech recognition,” IEEE Trans. Speech and Audio Processing, vol. 1, no. 1, pp. 59-67, January 1993.

[2] S. Ortmans, A. Eiden, H. Ney, and N. Coenen, “Look-Ahead techniques for fast beam search,” in Proc. ICASSP97, pp. 1783-1786, Munich, Germany, April 1997.

[3] H. Ney, R. Haeb-Umbach, B.H. Tran, and M. Oeder, “Improvements in beam search for 10000-word continuous speech recognition,” in Proc. ICASSP92, pp. 13-16, San Francisco, CA, March 1992.

[4] S. Renals, “Phone deactivation pruning in large vocabulary continuous speech recognition,” IEEE Signal Processing Letters, vol. 3, no. 1, January 1996.

[5] P.S. Gopalakrishnan, D. Nahamoo, M. Padmanabhan, and M.A. Picheny, “A channel bank based phone detection strategy,” in Proc. ICASSP94, pp. 161-164, Adelaide, Australia, April 1994.

[6] A. Rosenberg, J. Delong, C.H. Lee, B.H. Juang, and F. Soong, “The use of cohort normalized scores for speaker recognition,” in Proc. ICSLP92, pp. 467-470.

[7] M. Rahim, C.H. Lee, and B.H. Juang, “Discriminative utterance Verification for connected digits recognition,” IEEE Trans. on Speech and Audio Processing, vol. 5, no. 3, pp. 266-277, May 1997.

[8] T. Imai, A. Kobayashi, S. Sato, H. Tanaka, and A. Ando, “Speech recognition engine for real-time broadcast news captioning,” NHK Labs. Note, Serial No. 464, May 2000.

[9] L. Nguyen, and R. Schwartz, “Single-tree method for grammar-directed search,” in Proceedings of ICASSP’99, Phoniex, March 1999.

[10] Intel Architecture Software Developer’s manual 1999.

[11] W. Reichl, and W. Chou, “A decision tree state tying based on segmental clustering for acoustic modelling,” in Proceedings of ICASSP’98, May 1998.

[12] O. Siohan et al., “A Real-Time Japanese Broadcast News Closed-Captioning System,” Submitted to Eurospeech’01.