FAST IMPLEMENTATION METHODS FOR VITERBI-BASED WORD-SPOTTING

K. M. Knill *S. J. Young*
Cambridge University Engineering Department, Cambridge, CB2 1PZ, UK

ABSTRACT

This paper explores methods of increasing the speed of a Viterbi-based word-spotting system for audio document retrieval. Fast processing is essential since the user expects to receive the results of a keyword search many times faster than the actual length of the speech. A number of computational short-cuts to the standard Viterbi word-spotter are presented. These are based on exploiting the background Viterbi phone recognition path that is computed to provide a normalisation base. An initial approximation using the phone transition boundaries reduces the retrieval time by a factor of 5, while achieving a slight improvement in word-spotting performance. To further reduce retrieval time, pattern matching, feature selection, and Gaussian selection techniques are applied to this approximate pass to give a total $\times 50$ increase in speed with little loss in performance. In addition, a low memory requirement means that these approaches can be implemented on any platform, including hand-held devices.

1. INTRODUCTION

The message domain of many word-spotting applications, such as personal memo and dictation retrieval, tends to be very user-specific and liable to change over time. An open keyword vocabulary is therefore essential to allow the user to search for any term in the audio database. However, if an open keyword set is used, the location of keyword hits cannot be determined in advance of a retrieval request. Since the user expects to receive the results of a keyword search in a reasonably short time, the retrieval process must operate much faster than the actual length of the speech. For example, to achieve a response of 3 seconds for 1 minute of data the processing needs to be $20\times$ faster than real-time.

An approximation to a full Viterbi word-spotter with a network of the keyword and filler models in a loop is presented that reduces the amount of computation necessary, thereby, speeding up retrieval (section 3.). To further reduce retrieval time, feature selection, pattern matching, and Gaussian selection have been applied (sections 5. to 6.). Experimental results are presented in section 7., showing an overall speed improvement of $\times 50$. In addition to their speed advantage, these methods require little memory so they can be applied on any platform, including hand-held devices. This contrasts with other fast implementation approaches reported previously such as lattice-based word-spotting systems which have been shown to be very fast [3], but require a large amount of memory for lattice storage.

2. STANDARD WORD-SPOTTING SYSTEM

Two recognition passes are run in the standard word-spotting system. In the first, the keyword and filler models are run together to determine putative keyword hits. The filler models are also applied separately to allow the filler scores to be used to normalise the keyword scores [6]. Figure 1 shows the basic word-spotting system structure. A concatenated string of phone HMMs is used to represent the keyword, the keyword phone string. The full set of phone HMMs is used in parallel as filler models to represent non-keyword speech. The filler only recognition is effectively a Viterbi phone recogniser and it can be applied in advance when the message is recorded, as it is keyword independent, so that only the keyword plus filler recogniser has to be run when a search request is received.

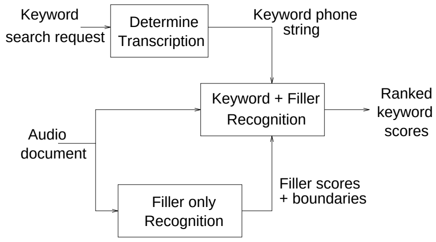

Figure 1. Word-spotting system structure

3. APPROXIMATION TO WORD-SPOTTING RECOGNITION PASS

There is a large amount of duplification of effort in computing the filler model hypotheses in a standard keyword and filler recogniser. This can be greatly reduced if instead of calculating the filler probabilities at any frame, the Viterbi filler hypothesis is used.

In word-spotting, only the score over the keyword frames is required, not the total path score, e.g. the score for a keyword between frames $f(1)$ and $f(2)$ is

$$\begin{aligned}\log l(keyword) \quad &= \quad \log l(o_{f(0)}, \ldots, o_{f(2)}|keyword) - \\ & \quad\quad \log l(o_{f(0)}, \ldots, o_{f(1)}|filler) \qquad (1)\end{aligned}$$

where $\log l(o_{f(0)}, \ldots, o_{f(1)}|filler)$ is the optimal filler path log-likelihood score up to frame $f(1)$. This has been calculated in the filler recogniser. If the addition of a keyword is assumed to not affect the path scores of keyword matches elsewhere in the same path, then only the keyword frame scores have to be calculated in the recogniser. If the forward and backward Viterbi path scores are stored at each time frame, O(2T) frames must be stored, giving $O(T^2)$ possible start and end points for each keyword [2].

To further reduce the memory requirements and computation, the assumption can be made that the phone transition boundaries in the word-spotting path are identical to those in the filler path. This assumption means that the score up to a transition frame, $t(1)$, is known and the only requirement is to calculate the score for the keyword starting at $t(1)$ and finishing at $t(2)$, where $t(2) > t(1)$ are phone boundaries in the filler path, as illustrated in figure 2. Since the scores and transitions are recorded at the phone level, the maximum number of transitions possible is of O(T/3) for 3 state phone models. Far fewer computations are, therefore, required in the word-spotter.

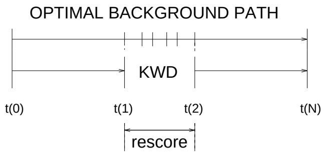

Figure 2. Faster keyword spotting recognition pass, single keyword case

Using the token passing paradigm [11], operation of the recogniser is as follows. At each filler model boundary, a token is propagated to the keyword model. The token contents are set to match the filler model path up to that frame. If any tokens are emitted by the keyword model at this point, the highest scoring token is recorded. The ratio scoring process, in which the keyword score is normalised against the background Viterbi pass to improve the rank order, can be performed immediately. The average keyword frame score is normalised by the average filler model score over the same set of frames (see figure 2). A threshold can then be applied to test if the keyword should be accepted. Once recognition is complete, overlapping keyword hits are eliminated by removing any keywords of a lower score that overlap with the best keyword and so on down the set of putative hits. A ranked list of keyword hits, based on the normalised score, is then made over all the audio documents in the search space and the information given to the user.

4. PATTERN MATCHING

In the above approximation the keyword recogniser is applied to all frames of speech data to be searched. If the recogniser is applied instead to a subset of the frames, then, an improvement in speed will result. Since the same model set is used in the keyword phone string and fillers, the phone label information from the Viterbi filler path can be used to determine which segments of the speech are likely to contain a keyword.

The recognised phone path can be scanned for matches, or partial matches, of the keyword phone string. The keyword recogniser is then only run over those frames that lie within matched segments. The simplest criterion to use for a match is to force the recognised string and keyword string to be identical. However, the number of matches found in this way would be very small due to recognition errors, so instead a partial match criterion is required.

Dynamic programming is used to perform the string matching. Penalties for substitution, deletion and insertion are used in the DP alignment algorithm. The penalties can either be fixed or be phone dependent. Only the former is investigated. For each starting point, the best DP alignment is stored provided at least one phone match between the two strings is recorded. Further thresholding can be applied based on the DP scores. To try and ensure that the number of keyword frames eliminated is kept to a minimum, the endpoints of the DP match can be extended by one or more transition points, at the cost of increasing the search space. To limit this increase a threshold can be applied to prevent extension.

Most of the computational effort remaining is expended computing the Gaussian output probabilities. The following techniques reduce the number of probability calculations, thereby further reducing the retrieval time.

5. FEATURE SELECTION

The keyword recognition pass can be speeded up by reducing the number of features in each acoustic vector, thereby, decreasing the number of Gaussian output probability calculations performed. Power spectrum resolution, the *F*-ratio, and mutual information have all been used to select the optimum set of features to maximise discrimination and recognition performance. However, manually selected sets have typically out-performed these methods [10], so are used here. Our standard system uses 39 element acoustic vectors consisting of static MFCCs, energy, deltas and accelerations. As an alternative, a reduced 26 feature set [10] has been used to improve the response time by a factor of 3. The set consists of

$$R(26) \quad = \quad C_{1-10}, E, \triangle C_{1-8}, \triangle E, \triangle^2 C_{1-5}, \triangle^2 E$$

This allows direct comparison with the standard 26 feature set of statics, log energy and delta parameters.

6. GAUSSIAN SELECTION

An alternative approach would be to store the N highest state likelihood scores per frame from the filler path. No likelihood computation would then need to be done in the word-spotting path. However, the memory cost of storing the likelihoods may grow quite large. A more efficient approach is to use Gaussian clustering combined with vector quantization.

It is well known that Gaussian models are statistically accurate only if the input feature vector is “near” to the Gaussian means. When the feature vector falls on the tail of a distribution, the Gaussian model provides at best a poor approximation of the likelihood. In this outlier case, a simple approximation to the likelihood maybe sufficient. So only the likelihoods of those Gaussian components for which the input vector is not an outlier need to be calculated exactly. To determine which Gaussians to use, all the mixture components in the model set are clustered into neighbourhoods. A vector quantizer is defined, with a codeword for each neighbourhood [1]. In the recogniser, each input feature vector is quantised. The likelihood of each Gaussian in the corresponding cluster is calculated exactly. A quick approximation is made to the likelihood of the other Gaussians, for example by table look-up or by a small constant.

The Gaussian clustering was performed during HMM training as follows. A weighted (Mahalanobis-like) Euclidean distance between the means was used to calculate the distance between the ith and jth Gaussians.

$$\delta(\mu_i, \mu_j) = \frac{1}{D} \sum_{k=1}^{D} \{ w(k)(\mu_i(k) - \mu_j(k)) \}^2 \tag{2}$$

where D is the feature vector dimension, $\mu_i(k)$ is the kth component of vector μ_i, and $w(k)$ is equal to the inverse square root of the kth diagonal element of the average of the covariances of the Gaussian set $G(\mu_m, \mathbf{\Sigma}_m), m = 1, \ldots, M$ where $\mathbf{\Sigma}_m$ is the diagonal covariance of the mth Gaussian component. Then, a clustering procedure based on the Linde-Buzo-Gray algorithm was applied to minimise the average (per mixture component) distortion, δ_{avg},

$$\delta_{avg} = \frac{1}{M} \sum_{m=1}^{M} \left\{ \min_{\phi=1}^{\Phi} \delta(\mu_m, \mathbf{c}_\phi) \right\} \tag{3}$$

where M is the number of Gaussian components in the model set, Φ is the number of clusters (pre-defined), and $\mathbf{c}_\phi$ is the centre (codeword) of the ϕth cluster χ_ϕ,

$$\mathbf{c}_\phi = \frac{1}{size(\chi_\phi)} \sum_{m \in \chi_\phi} \mu_m, \phi = 1, \ldots, \Phi \tag{4}$$

The clusters produced by the above process are disjoint. If used in recognition errors are likely, as the likelihood of some Gaussians close to the input feature vector but not in the selected cluster will not be exactly computed. Instead clusters are allowed to share Gaussians as follows. Given a threshold $\Theta \gg 1$, an input feature vector, $\mathbf{o}$, is said to fall on the tail of the mth Gaussian if

$$\frac{1}{D} \sum_{i=}^{D} \frac{(o(i) - \mu_m(i))^2}{\sigma_m^2(i)} > \Theta \tag{5}$$

where $\sigma_m^2(i)$ is the ith diagonal element of $\mathbf{\Sigma}_m$. Thus, if the cluster centroid is taken to be a typical input feature vector the neighbourhood, ν_ϕ, of codeword $\mathbf{c}_\phi$, can be defined as consisting of the Gaussians such that [1]

$$G(\mu_m, \mathbf{\Sigma}_m) \in \nu_\phi \;\; iff \;\; \frac{1}{D} \sum_{i=1}^{D} \frac{(c_\phi(i) - \mu_m(i))^2}{\sigma_{avg}^2(i)} \leq \Theta \tag{6}$$

where $\sigma_{avg}^2(i)$ is the ith diagonal element of the matrix of the average covariance of the full Gaussian set. The use of $\sigma_{avg}^2(i)$ in the criterion is preferred to $\sigma_m^2(i)$ since the individual variance estimates are often noisy.

The *weighted* distance measure in equation 6 takes no account of the variance of the cluster, so some Gaussians may be incorrectly assigned. An alternative, *class-weighted*, distance measure was therefore implemented, where the neighbourhood of codeword, $\mathbf{c}_\phi$, consists of the Gaussians such that

$$G(\mu_m, \mathbf{\Sigma}_m) \in \nu_\phi \;\; iff \;\; \frac{1}{D} \sum_{i=1}^{D} \frac{(c_\phi(i) - \mu_m(i))^2}{\sqrt{(\sigma_{avg}^2(i)\sigma_{c_\phi}^2(i))}} \leq \Theta \tag{7}$$

where $\sigma_{c_\phi}^2(i)$ is the ith diagonal element of the cluster centroid covariance, $\mathbf{\Sigma}\mathbf{c}_\phi = 1/size(\chi_\phi) \sum_{m \in \chi_\phi} \mathbf{\Sigma}_m$.

The choice of Θ controls the average size of the Gaussian neighbourhoods. Efficiency improves with reductions in Θ because fewer Gaussian likelihoods have to be computed during recognition. However, there is a trade-off with the recognition accuracy.

During recognition, the appropriate neighbourhood is selected by determining the cluster centroid, $\mathbf{c}_i$, which minimises the weighted (Mahalanobis-like) Euclidean distance to the observation vector, $\mathbf{o}(t)$, at time t,

$$\mathbf{c}_i = \min_{\phi=1}^{\Phi} \delta(\mathbf{o}(t), \mathbf{c}_\phi) \tag{8}$$

The reduction in computation of the Gaussians, the computation fraction, C, is therefore

$$C = \frac{G_{new} + VQ_{comp}}{G_{full}} \tag{9}$$

where G_{new} is the average number of Gaussians calculated per frame in the selection system, G_{full} the total number of Gaussians, and VQ_{comp} the number of computations required to calculate the VQ index. In the word-spotting system, the VQ computation is carried out in the filler path and the VQ indices stored with the MFCCs. This allows more Gaussians to be calculated for the same computation fraction in the word-spotting pass, as $VQ_{comp} = 0$.

7. EXPERIMENTAL RESULTS

7.1. System description

The Video Mail Retrieval Database (VMR1) [4] was used for evaluation. The HMMs for each speaker were trained on the read sentences ($\sim$ 200). Testing was carried out on 20 spontaneous speech messages from the same speaker.

46 speaker dependent, continuous density multiple component Gaussian distribution, monophone HMMs were used. Each model had 3 emitting states, with 8 mixture components per state, with a left-to-right topology, no skips, except for the silence and pause models, both of which were ergodic with 3 and 1 states respectively. The parameters used in the system were; 12 Mel-frequency Cepstral Coefficients, normalised log energy, and first and second derivatives of all parameters, giving a vector size of 39, unless otherwise stated. The monophone HMMs were used in the word-spotter for both the background filler models and the sub-word units in the keyword models. The keyword phone transcriptions were taken from the British English Pronunciation (BEEP) dictionary [9]. For Gaussian selection, the likelihoods for Gaussians outside the VQ cluster were approximated by a constant log likelihood value of -500.0.

Putative keyword hits were re-scored by dividing the maximum log likelihood keyword score by the average filler model score over the same time frames. This has been shown to yield a better keyword ranking than other proposed schemes [6]. Results are averaged over the 15 speaker set. The results are presented for acoustic hits, i.e. where the phone sequence matches that of the keyword.

All training and testing used version 1.5 of the HTK HMM toolkit [12], with suitable extensions to perform word-spotting and implement the various speed-up techniques.

7.2. Results

The results in table 1 show that a $\times 5$ reduction in computation time is achieved by using the keyword only recogniser (Keyword) compared to the full keyword and filler recogniser (Full). A slight improvement in performance is also observed. Applying DP with fixed penalties and requiring a minimum of one hit per frame set (Fixed DP) gives an extra 12% drop in retrieval time. However, the word-spotting accuracy is severly degraded. This is chiefly due to frames at the start and end of keywords being eliminated by the search, lowering the keyword hit score. To overcome this, the neighbouring transition boundary was added to each end of the DP frame sets when the number of hits was less than the length of the keyword phone string (Fixed Extd DP). Using this DP match, the word-spotting

System	No. of False Alarms			Rel
	1	3	1-10	Time
Full	64.3	81.0	82.7	1.000
Keyword	66.2	82.7	84.9	0.200
Fixed DP	35.6	44.2	47.5	0.080
Fixed Extd DP	65.3	81.9	84.2	0.140
Fixed Extd2 DP	65.5	81.4	83.8	0.030
Fixed DP, No Acoust	50.0	57.9	60.6	0.001
Std 26 Features	62.8	78.4	80.6	0.133
Sel 26 Features	63.3	77.8	80.0	0.133
GS Nc64 Weight	64.4	80.7	83.1	0.133
GS Nc64 ClassWt	65.2	81.2	83.6	0.133
GS Nc128 ClassWt	64.8	81.0	83.6	0.133
GS Nc64 ClassWt DP	64.9	81.0	83.1	0.020
GS Nc128 ClassWt DP	65.1	80.8	83.2	0.020

Table 1. Word-spotting performance (% hits per false alarm) and relative retrieval times

performance of the full system is achieved, but computation time is increased. By adding the constraint that at least 2 keyword phone matches must be found in a frame set (Fixed Extd2 DP), word-spotting performance is maintained at the full level, while reducing the computation time to 1/30th. When the DP match is used on its own (Fixed DP, No Acoust) a much faster response is achieved. However, the word-spotting performance is once more severely degraded, showing the need for a keyword recogniser.

If the accelerations are not used, the acoustic vector reduces from 39 to 26 features (Std 26 Features). This gives a 1/3rd reduction in the computation time in the keyword recognition time, but a drop in word-spotting performance is observed. The latter was expected to be improved by using a selected set of 26 features. However, a slightly poorer performance is observed using the 26 element set defined in section 5. (Sel 26 feature).

For Gaussian selection, 64 weighted (GS Nc64 Weight), and 64 and 128 class-weighted (GS Nc64 ClassWt, GS Nc128 ClassWt), clusters were investigated. In all cases, it was found that the number of likelihood calculations could be reduced to 50% with little loss in word-spotting performance. Table 1 gives the performance for $\Theta = 2.2$, where 49.8, 47.1% and 42.3% gaussians were computed for the 64 weighted, and 64 and 128 class-weighted cases respectively. Hence, a 1/3rd reduction in retrieval time is possible, as for feature selection, but with little degradation in performance. As fewer Gaussians were computed per frame, the word-spotting performance dropped. A slower rate of degradation was seen using a class-weighted compared to a weighted tail threshold. Increasing the number of clusters further slowed the drop in performance. For example an average of 81.4% hits per false alarm were achieved using 30.5% Gaussians and 128 clusters, compared to 76.2% using 28.1% Gaussians and 64 class-weighted clusters, and 74.0% using 28.7% Gaussians and 64 weighted clusters. Similar behaviour was observed when Gaussian selection was added to the pattern matching system, Fixed Extd2 DP, (GS Nc64 ClassWt DP, GS Nc128 ClassWt DP). This combined system reduced the retrieval time to 1/50th of the full system.

Gaussian selection also reduces the computation required for the filler path. For a particular tail threshold, Θ, the proportion of likelihoods that have to be computed exactly is less than that needed in the word-spotting path. With $\Theta = 2.2$, computation fractions of $C = 0.43$ and 0.50 were required for 64 and 128 class-weighted clusters, respectively.

8. CONCLUSIONS AND FURTHER WORK

Low cost approaches to open vocabulary word-spotting have been presented. These stemmed from an approximation to the full word-spotting pass using a precomputed filler pass, the keyword only recogniser. The basic method increased the retrieval speed $\times 5$, and yielded a slight improvement in word-spotting performance. Pattern matching, feature and Gaussian selection were then applied to increase the speed further. Experimental results showed that by adding pattern matching and Gaussian selection to the keyword only recogniser a $\times 50$ reduction in time is possible with little loss of word-spotting performance.

9. ACKNOWLEDGEMENTS

This work was funded by Hewlett Packard Laboratories, Bristol.

REFERENCES

[1] Bocchieri, E. *Vector quantization for efficient computation of continuous density likelihoods*, Proc ICASSP'93, Minneapolis, 1993.

[2] Huang, E-F., Wang, H-C., and Soong, F.K. *A fast algorithm for large vocabulary keyword spotting application*, IEEE Trans on Speech and Audio Processing, 2(3), pp 449-452, July, 1994.

[3] James, D.A. and Young, S.J. *A fast lattice-based approach to vocabulary independent wordspotting*, Proc ICASSP'94, Adelaide, 1994.

[4] Jones, G. J. F., Foote, J. T., Sparck Jones, K., and Young, S. J. *Video Mail Retrieval Using Voice: Report on Keyword Definition and Data Collection (Deliverable Report on VMR Task No 1)*, University of Cambridge Computer Laboratory, Tech. Report No. 335, May, 1994.

[5] Jones, G.J.F., Foote, J.T., Sparck Jones, K., and Young, S.J. *Video Mail Retrieval: the effect of word spotting accuracy on precision*, Proc ICASSP'95, Detroit, 1995.

[6] Knill, K. M. and Young, S.J. *Speaker Dependent Keyword Spotting for Accessing Stored Speech*, Cambridge University Engineering Dept., Tech. Report No. CUED/F-INFENG/TR 193, 1994.

[7] Linde, Y., Buzo, A. and Gray, R.M. *An Algorithm for Vector Quantizer Design*, IEEE Trans. on Comms., Vol COM-28, No. 1, pp84-94, Jan, 1980.

[8] Murveit, H., Monaco, P., Digilakis, V. and Butzberger, J. *Techniques to achieve an accurate real-time large-vocabulary speech recognition system*, Proc ARPA Workshop on Human Language Technology, Plainsboro, N.J., pp368-373, Mar, 1994.

[9] Robinson, T., Fransen, J., Pye, D., Foote, J., and Renals, S. *WSJCAM0: A British English speech corpus for large vocabulary continuous speech recognition*, Proc ICASSP'95, Detroit, 1995.

[10] Valtchev, V. *Discriminative Methods in HMM-based Speech Recognition*, PhD Thesis, Cambridge University, 1995.

[11] Young, S. J., Russell, N. H., and Thornton, J. H. S. *Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems*, Cambridge University Engineering Department, Tech. Report No. TR.38, July, 1989

[12] Young, S. J. Woodland, P. C., and Byrne, W. J. *HTK: Hidden Markov Model Toolkit V1.5*, Entropic Research Laboratories Inc., 1993.