Speaker Dependent Keyword Spotting for Accessing Stored Speech

K. M. Knill & S. J. Young

CUED/F-INFENG/TR 193

October 1994

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Email: kmk@eng.cam.ac.uk

Abstract

This report investigates the use of a speaker-dependent HMM word-spotter to retrieve spoken messages. The baseline word-spotter consists of a parallel network of keyword and background filler models. A further pass, using the filler models only, can be used to re-score the putative keyword hits. Word-spotting performance using (i) whole-word and (ii) sub-word keyword models is investigated. For each keyword model type, effects on performance of re-scoring, Gaussian component number, and parameter set are evaluated on a set of spoken messages, taken from the Video Mail Retrieval database. Overall, sub-word models are shown to yield a higher hit rate, particularly before the first false alarm occurs.

Keywords: word-spotting, speech recognition, information retrieval.

Contents

1 Introduction

There is a growing interest in the use of hand-held computing devices for a wide range of applications. As dimensions decrease, the use of voice as an input media becomes more attractive. A typical application being studied is "Voice Notes" which allows short spoken documents to be stored and subsequently retrieved by category or content.

Using audio to enter and retrieve data allows smaller hand-held devices to be designed, for example Stifelman's VoiceNotes [8], and the recently launched product Voice Organizer (Voice Powered Technology International Inc., 1993). In the extreme case, the physical interface may be negligible, requiring only a speaker and microphone [8]. The audio medium can also have advantages over handwritten or typed records in capturing information. It can be a faster means of capture, and more expressive. However, it is correspondingly more difficult to review information in audio, rather than visual form. Speech is slow to access, and cannot be scanned easily, in the way that visual cues are used to search pages of text for a specific item. To retrieve documents based on their audio content, word-spotting can be used. Unlike a continuous speech recogniser which attempts to match every word, a word-spotter is only interested in determining the presence and position of specific speech segments (keywords) within each record. The specified segment is typically a word or phrase, for example "find EUROSPEECH PROCEEDINGS", but in general may also be a non-linguistic sound.

Word-spotting systems based on Hidden Markov Models (HMMs) are considered in this report. These have proved more successful at modelling arbitrary speech than template based systems [5], [9]. For each keyword, an HMM is trained using statistical estimation techniques. Non-keyword speech is modelled by one or more HMMs. Word-spotting is performed by running a continuous speech recogniser with the keyword and non-keyword models, and the recogniser outputs the sequence of HMMs which best models the unknown speech data. Putative keyword hits are ranked according to their score (defined in section 2). If the keyword is present at that location, this is called a true hit, otherwise it is a false alarm. Performance is measured in terms of the number of true hits achieved for a given false alarm rate.

Two classes of recognition systems have been considered, based on (i) whole-word, and (ii) subword keyword models. These are presented in section 2. The work has been based on the assumption that the class of device it is aimed at is for personal use. Hence, a speaker-dependent word-spotting approach has been taken, which assumes that the word-spotting HMMs are derived from the same speaker as the data being searched. This will not always hold as the speaker may record phone calls, conversation, and other transactions which violate this assumption. The database used for training the HMMs and testing the word-spotting system is described in section 3. The keyword and filler models, and the performance measure used, are defined in sections 4 and 5 respectively. Results of the word-spotting tests performed are given in section 6. Future work areas are proposed in section 7.

2 Recognition systems

The baseline recognition system was built using the HTK Version 1.5 toolkit [11]. This system is a speaker dependent, null-grammar, time synchronous Viterbi beam search decoder consisting of a parallel network of keywords and fillers, with silence enforced at the start and end of each sentence (figure 1). Whole-word continuous density mixture Gaussian distribution HMMs are used to model the keywords, trained on keyword speech. Following Rose and Paul [5] a set of 43 monophone models are used as background filler models, trained on non-keyword speech.

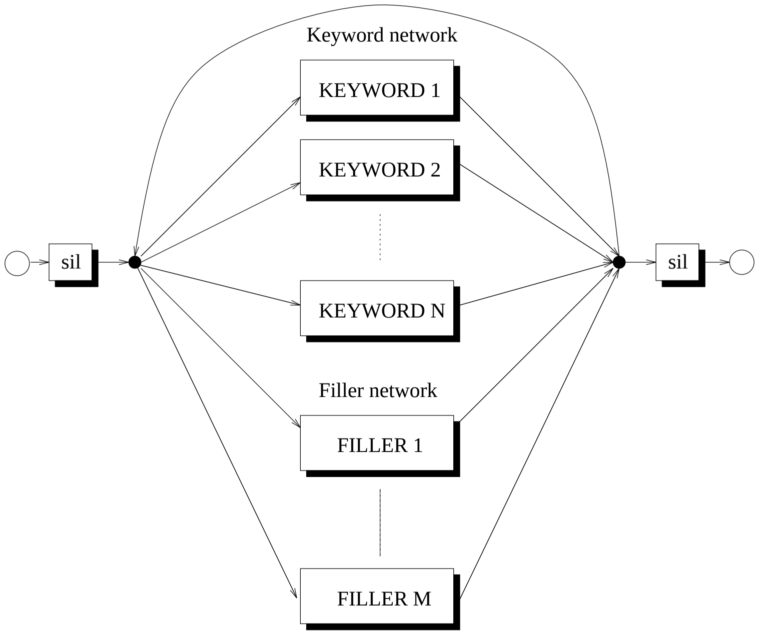

Figure 1: Baseline null-grammar word-spotting network

Recognition is performed using the token passing paradigm, where the best path to a particular instance in the network is described by a token held in that state [10]. When a token is propagated externally from model M_i to model M_j the log probability of that token is incremented by

$$s \log[P(M_j|M_i)] + p \tag{1}$$

where s is a grammar scale factor, p a fixed transition penalty, and

$$P(M_j|M_i) = 1/N_{succ(i)} \tag{2}$$

where $N_{succ}(i)$ is the number of successors to model M_i. Since all models follow one another in the baseline system, their propagation weighting is the same.

As the keyword likelihood scores often exhibit variability in time, assigning reliable decision regions for separating true keyword hits from false alarms is difficult. Rose and Paul [5] showed that significant improvement can be achieved if Likelihood Ratio scoring is applied. In their work a second recognition pass is performed using a network consisting of the background filler models alone. A modified keyword log likelihood score can then be computed

$$S_{LR} = S_{KW} - S_{BA} \tag{3}$$

where S_{KW} is the Viterbi maximum likelihood score per frame for the keyword,

$$\begin{aligned} S_{KW} &= \frac{\text{KW Model Exit log prob - KW Model Entry log prob}}{\text{number of frames KW model recognised over}} \\ &= \frac{\max_S \{\log [\, P(O, S|KW)\,]\}}{NF_{KW}} \end{aligned} \tag{4}$$

where the keyword start and end frames are given by f_s, f_e, respectively, O is the sequence of observations, $\{o_{f_s}, \ldots, o_{f_e}\}$, and $NF_{KW} = f_s - f_e$ is the number of frames that the keyword is recognised over. S_{BA} is the average maximum likelihood score per frame for the overlapping string of background fillers (see figure 2),

$$S_{BA} = \frac{\sum_{i=1}^{m} \left(S_{(BA,i)} NF_{(BA,i)} \right)}{NF_{KW}} \tag{5}$$

where $S_{(BA,i)}$ is the Viterbi maximum likelihood score of the ith background model,

$$S_{(BA,i)} = \frac{\max_S \{\log [\, P(O, S|(BA,i)) \,]\}}{NF_{(BA,i)}} \tag{6}$$

and $NF_{(BA,i)}$ is the number of frames of the ith background model that overlap the keyword frames.

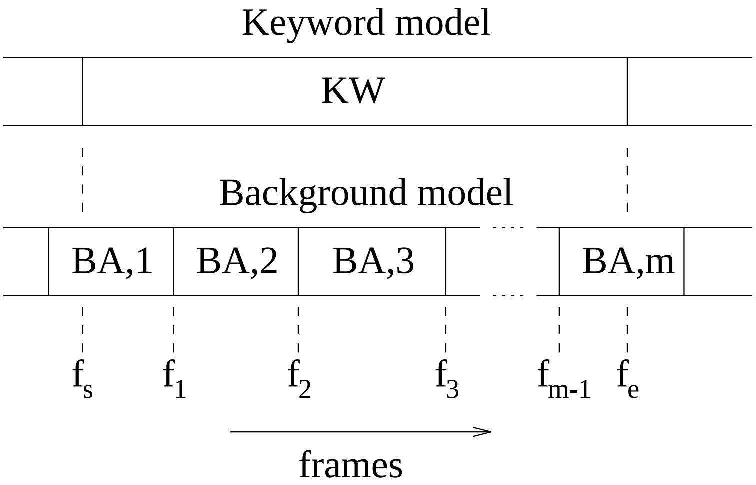

Figure 2: Keyword and background filler models recognised over the same time frames, using the keyword plus filler network and the filler only network respectively

Given the decoded keyword start and end times, t_s, t_e respectively, Rose [7] suggested an alternative re-scoring mechanism, MS_{LR}, based on the maximum difference between the log keyword probability and the log filler probability over the interval $t_s \leq t \leq t_e$

$$MS_{LR} = \max_t \left(\log P(O_{t_s}^t | KW) - \log P(O_{t_s}^t | BA) \right) \tag{7}$$

where $\max_t \log P(O_{t_s}^t | KW)$ represents the maximum log probability for the keyword model, KW, over the observations $O_{t_s}^t = o_{t_s}, \ldots, o_t$, and $\max_t \log P(O_{t_s}^t | BA)$ represents the maximum filler network probability decoded over the same observations. Rose showed that the score MS_{LR} is an approximation to the *a posteriori* keyword log probability. As the HTK Viterbi recogniser tool HVite outputs the average maximum likelihood per frame score for each HMM, MS_{LR} is approximated by finding the maximum difference in log keyword and filler probabilities over the frame interval, $f_s \leq f \leq f_e$. The keyword model score at frame f is assumed to be $S_{KW}(f) = (f - f_s)S_{KW}$, and the background model score to be

$$S_{BA}(f) = (f_1 - f_s)S_{(BA,1)} + (f_2 - f_1)S_{(BA,2)} + \ldots + (f - f_{(n-1)})S_{(BA,n)} \tag{8}$$

Then MS_{LR} becomes

$$MS_{LR} = \max_f \left(S_{KW}(f) - S_{BA}(f)\right) \tag{9}$$

However, it has been found that better distinction between false alarms and true hits can be achieved by using a normalisation type re-scoring mechanism, NS_{LR}, defined as

$$NS_{LR} = \frac{S_{BA}}{S_{KW}} \times 100 \tag{10}$$

If the keyword score is better than the background model score, i.e. it is closer to zero, then $NS_{LR} > 100$, else $NS_{LR} \leq 100$. In general, true hits should have higher scores than false alarms, and will, therefore, be placed at or close to the top of an ordered list of putative hit scores. Thresholding can then be applied to remove most of the false alarms.

Rescoring method	No. of False Alarms			
	1	2	3	10
S_{LR}	50.0	64.3	68.1	80.2
MS_{LR}	43.4	55.5	58.8	73.6
NS_{LR}	50.5	65.4	69.2	78.6

Table 1: % Hits per false alarm using (i) likelihood ratio, S_{LR}, (ii) maximum difference, MS_{LR}, and (iii) normalisation, NS_{LR}, rescoring methods

Table 1 shows the percentage of total true hits before the nth false alarm achieved using S_{LR}, MS_{LR}, NS_{LR}. The null-grammar word-spotting network consisted of whole-word keyword models and a background silence model, each of 2 component mixtures, with static plus delta coefficients. From table 1, NS_{LR} can be seen to achieve the best performance overall, although it is not well-founded theoretically unlike S_{LR}, MS_{LR}. The improvement is more pronounced when sub-word keyword models are used. A possible reason for this improvement is the fact that the absolute keyword score is not taken into account in the S_{LR} and MS_{LR}, while NS_{LR} does to a limited extent, e.g.

$$\begin{aligned} S_{KW} = -60, S_{BA} = -63 \quad &\rightarrow \quad S_{LR} = 3, NS_{LR} = 105 \\ S_{KW} = -75, S_{BA} = -78 \quad &\rightarrow \quad S_{LR} = 3, NS_{LR} = 104 \end{aligned}$$

Combining the keyword score in some way with either S_{LR} or MS_{LR} may give better word-spotting performance. This is not considered in this report, NS_{LR} is used throughout.

Further improvements in performance were found to be achieved when the filler model component of the word-spotting network shown in figure 1 was reduced to just the silence model. The results for the keyword and silence network are normalised against those for the full filler model network. This system is known as the *normalised whole-word system*. For example, table 2 shows the performance of the normalised keyword and filler network against that of the normalised whole-word system for two component mixture, static plus delta parameter HMMs.

Word-spotting network	No. of False Alarms			
	1	2	3	10
normalised keyword + filler	50.5	59.3	61.5	61.5
normalised keyword + silence	50.5	65.4	69.2	78.6

Table 2: % Hits per false alarm of rescored (a) keyword + filler, (b) keyword + silence networks, using whole-word keyword models

In a fully open system, the keywords should be user-definable, and not necessarily in the dictionary [6], or the initial training data. Pre-training of the keyword models, as in the whole-word systems above, will therefore not be possible. To start to address this issue, a sub-word approach, which does not distinguish between keyword and non-keyword speech in training, was investigated.

In the sub-word approach the set of monophone filler models are trained on both keyword and non-keyword speech. The keyword models are built by concatenating the monophone models. When a true hit is scored, the keyword phone sequences have a lower log likelihood score than the equivalent string of background phones due to the difference in word-internal and word-external inter-model log transition probabilities. For word-internal nodes, the inter-model log transition probability is $-\log[N_{succ}(i)]$. As a maximum of 2 models follow any word-internal keyword node, the word-internal transitions have a higher probability than the corresponding word-external transitions.

Normalisation was again found to improve performance. In this case, however, using the full set of monophones as the filler models in parallel with the keyword models (as in figure 1) was seen to give better performance than using just the silence model. The normalised keyword plus filler network, known as the *normalised sub-word system*, was therefore used to generate the sub-word keyword model results presented in this report. For example, table 3 shows the performance of the keyword plus filler, normalised keyword plus filler, and the normalised keyword plus silence networks, for four component mixture, static plus delta parameter HMMs.

Word-spotting network	No. of False Alarms			
	1	2	3	10
keyword + filler	32.4	50.5	64.8	79.1
normalised keyword + filler	59.3	67.0	70.9	80.8
normalised keyword + silence	51.1	60.4	64.3	76.9

Table 3: % Hits per false alarm of (a) keyword + filler, (b) rescored keyword + filler, and (c) rescored keyword + silence networks, using sub-word keyword models

3 Database

The Video Mail Retrieval (VMR) Database 1 [1] was used. This database has been designed for word-spotting and information retrieval in spontaneous speech. It consists of a set of acoustic training data and a set of spontaneous messages for testing, collected from 15 speakers (11 male, 4 female). The messages were generated for a series of prompting scenarios built round a set of 35 keywords, which constitute the initial search vocabulary for retrieval experiments. The experiments reported here use the data recorded by speaker 3 (female).

The acoustic training data consists of: a set of isolated utterances (5 off) of each keyword; a collection of 64 carrier phrases with at least 5 examples of each keyword; 12 read sentences containing at least 2 examples of each keyword; and a set of 150 read sentences from the phonetically rich TIMIT database [3] for training the acoustic filler models.

Each speaker recorded 20 spontaneous messages, relating to 4 categories. These were selected, from a set of 10, to cover topics that the speaker was familiar with. Speaker 3 recorded messages from the the spotting, output, windows and equipment categories. The use of a sub-set of scenarios results in a skewed distribution of keywords for each speaker, as shown in table 4 for speaker 3. This, however, is a realistic scenario since the user may have either erased, or not yet recorded, messages containing certain keywords.

Keyword	No.	Keyword	No.	Keyword	No.
active	1	location	0	retrieve	5
assess	3	mail	17	score	5
badge	0	manage	3	search	9
camera	0	meeting	0	sensor	0
date	2	message	21	spotting	6
display	3	microphone	6	staff	2
document	1	network	2	time	12
find	7	output	5	video	12
indigo	5	pandora	0	windows	11
interface	4	plan	0	word	14
keyword	15	project	0	workstation	6
locate	0	rank	5		

Table 4: Distribution of keywords in test messages for speaker 3 in VMR DATABASE 1

4 Keyword and filler models

Each system contained 35 keyword and 43 monophone models. The keyword models consisted of a varying number of emitting states, defined below, and the monophones had 3 emitting states, each with a continuous density mixture Gaussian output distribution. Each model had a left-to-right topology with no skips. The training data was preprocessed using a 25 msec Hamming window and a 10 msec frame period. Additionally the data was pre-emphasised with a factor of 0.97 and liftered with a factor of 22. Each frame was coded into a parameter vector of 12 mel-frequency cepstral coefficients and the normalised log energy. The first (delta coefficients) and second (acceleration coefficients) time differentials were added to make 26 and 39 element vectors respectively.

For each parameterisation, the tool HCompV was run on all the training data to generate a single overall variance vector to provide a variance floor macro for the diagonal elements of the state pdf covariance matrices. The variance floor macro was used in all the model estimation routines, after dividing each vector element by 100. Initially, single component mixture models were created.

HInit was used to establish the initial HMM parameter values. The tool HRest was then used to individually re-estimate the parameters of each HMM. The set of models were further re-estimated by using HERest to perform 4 cycles of embedded re-estimation. Multiple component mixture models were built in two stages. The number of component mixtures in the set of models was incremented by one using the MixUp command in the tool HHEd. Then, 4 cycles of embedded re-estimation were performed using HERest.

4.1 Whole-word

The whole-word keyword models were trained on the isolated and carrier speech in VMR DATABASE 1 (between 9 and 18 examples of each keyword). Three model classes were compared,

1. *fixed* - each model consists of 15 emitting states
2. *dict* - each model consists of 3 emitting states per phoneme, where the number of phonemes is given by a dictionary being developed at CUED based on the Advanced Oxford Learner's Dictionary[1]. The models were made up of between 9 and 27 emitting states, with an average of 17 states per model.
3. *frame* - assuming a dictionary is not available, the number of states in each model is determined from the data. The number of frames per keyword in isolated speech is approximately double that of read and spontaneous speech so the (read) carrier phrases were used to determine the average frame length of each keyword. The TIMIT data was used to determine the average frame length of a 3 emitting state monophone. Then, the number of states per keyword model is defined as

$$\text{number of states} = \frac{\text{average no. of frames per keyword}}{\text{average no. of frames per state per phone}} \tag{11}$$

   As for the *dict* class, the models were made up of between 9 and 27 emitting states, with an average of 17 states per model.

The background monophone models were trained on all non-keyword speech in the isolated word and read (carrier and TIMIT) sentences.

4.2 Sub-word

The background monophone models were trained on both keyword and non-keyword speech. All isolated and read speech in VMR DATABASE 1 was used. The keyword models were then built by linking the monophone models according to the keyword's pronunciation rules in the CUED dictionary.

5 Performance measure

The evaluation is done as follows. The putative keyword hits are first ordered by score from best to worst across all messages for each individual keyword. Then a tally is made of the number of true hits found as the 1st, 2nd, etc false alarm for each keyword is encountered. This corresponds to the number of keywords which would be found if the detection threshold were set at the score

[1]The original dictionary was obtained from ftp: black.ox.ac.uk

of each false alarm in turn. At each false alarm level, the tallies are added across keywords and expressed as a percentage of the total number of keyword examples (i.e. possible true hits) in the test data. For the ith false alarm,

$$\%\ \text{Hits} \ /\ i\text{th FA} = \frac{\sum_{j=1}^{m} NH_{i,j}}{NH} \times 100 \tag{12}$$

where m is the number of individual keywords, $NH_{i,j}$ is the number of hits scored for the jth keyword before the ith false alarm, and NH is the total number of true hits.

In an application such as VoiceNotes a maximum of three false alarms is expected to be tolerable. The number of hits at the 1st, 2nd and 3rd false alarm is therefore recorded. Since these statistics tend to have a large variance over test data, the number of hits at the 10th false alarm level is also noted.

The NIST figure of merit calculation [4] was not used as it measures the average performance over the 1st to 10th false alarms. It is also very dependent on the duration of the test data available, and having a more even distribution of the keywords than was found in these tests.

6 Results

The full set of 20 spontaneous messages for speaker 3 were used in each test. The word-external inter-model log transition probability (equation 1) had a grammar scale factor of 5, and a fixed transition penalty of zero.

6.1 Whole-word tests

6.1.1 Comparison of baseline and normalised whole-word systems

Table 5 shows that the normalised whole-word system yields significant performance improvements over the baseline system. Performance is only degraded with the static coefficients, in the 1 component mixture case up to the 10th false alarm, and the 3 component mixture case at the 1st false alarm level. In general, the improvement in performance becomes more significant as the number of false alarms increases.

Co-articulation effects proved to be the main reason for 'missing' keywords in the data. In addition, 'WORD' was detected as 'KEYWORD' several times as a result of co-articulation effects and the similarity between the two keywords. 50% of the misses occurred in the first set of spontaneous messages. This was largely due to the speaker having a cold at this session which affected the speech quality.

System	Param	No. of mix	No. of False Alarms 1	2	3	10
baseline	static	1	42.3	56.6	58.8	60.4
		2	44.0	52.7	56.0	57.1
		3	44.5	53.8	55.5	56.6
	delta	1	40.7	55.5	62.6	67.6
		2	38.5	59.3	61.0	61.5
		3	43.4	57.7	57.7	58.2
	accel	1	33.5	54.9	58.8	64.8
		2	41.2	57.1	58.2	59.9
		3	45.6	54.9	56.6	56.6
normalised	static	1	41.8	53.8	57.7	68.7
		2	44.0	56.6	59.9	68.7
		3	41.8	54.4	55.5	68.7
	delta	1	45.6	65.9	70.3	78.0
		2	50.5	65.4	69.2	78.6
		3	50.0	67.0	72.5	80.2
	accel	1	42.3	61.5	68.1	78.6
		2	51.1	68.1	72.5	79.1
		3	58.8	67.0	71.4	79.1

Table 5: % Hits per false alarm of (i) baseline, and (ii) normalised whole-word systems for various parameterisations, using *dict* state class whole-word models

6.1.2 Comparison of whole-word state class models

From table 6 it can be seen that the *dict* state class models, in general, achieve the best performance. This implies that varying the length of the keyword models is beneficial. The *fixed* state class of models tended to out-perform the *frame* state class models. However, the results show that estimating the whole-word model length from the data is feasible. Reducing the number of states of some of the keyword models in the *frame* and *dict* classes would probably improve performance, since some of the model lengths were possibly too long (up to 27 states) to enable a good keyword model to be built.

No of cpts	model class	No. of False Alarms			
		1 FA	2 FA	3 FA	10 FA
1	dict	45.6	65.9	70.3	78.0
	fixed	49.5	61.5	66.5	78.0
	frame	47.3	59.3	65.9	76.9
2	dict	50.5	65.4	69.2	78.6
	fixed	50.5	64.3	70.3	78.0
	frame	47.3	63.7	69.2	78.6
3	dict	50.0	67.0	72.5	80.2
	fixed	57.1	66.5	70.9	77.5
	frame	57.7	68.7	71.4	75.8

Table 6: % Hits per false alarm of (i) *dict*, (ii) *fixed*, and (iii) *frame* state class whole-word models in the normalised whole-word system, for 1, 2, and 3 component, delta parameter, models

6.1.3 Effect of number of components and parameterisation

Table 7 details the full set of results for the normalised whole-word system. The performance is very inconsistent, with different systems achieving the highest percentage of hits at each false alarm level. To give a clearer indication of the effect on performance of varying the number of component mixtures and changing the parameterisation, the results for the *dict*, *fixed* and *frame* state classes were averaged. Tables 8, 9 and 10 show the averaged results for the 1, 2 and 3 component mixture model cases with static, delta and acceleration coefficient parameterisation.

In all the mixture component cases, adding delta coefficients yields a significant improvement in performance over the static parameter models. No advantage is gained by adding acceleration coefficients in the 1 component mixture case, and only improves performance at the 1st false alarm level for the 3 component mixture case. Adding acceleration coefficients does help in the 2 mixture component case, particularly at the 1st (+4.8%) and 2nd (+3.6%) false alarm level.

For each parameterisation, performance is improved at every false alarm level by increasing the number of component mixtures from 1 to 2. No benefit is obtained by increasing the number of component mixtures to 3 with both improvements and degradation in performance being observed, depending on the parameterisation and false alarm level.

The large variabilty in results is due to the limited amount of training data available for the keyword models (9 to 18 examples) leading to poorly trained models. The monophone models are, in contrast, well trained with between 70 and 300 examples of each phone in the training data. This is also the reason why increasing the number of component mixtures from 2 to 3, and adding acceleration coefficients in general degrades performance. The results for the sub-word case, where larger amounts of training data were available, are far more consistent (see next section).

Param	Model class	No. of mix	No. of False Alarms			
			1	2	3	10
		1	41.8	53.8	57.7	68.7
	dict	2	44.0	56.6	59.9	68.7
		3	41.8	54.4	55.5	68.7
		1	46.2	57.1	59.9	69.8
static	fixed	2	46.7	58.8	62.6	70.9
		3	45.6	59.9	64.8	72.0
		1	45.1	58.2	59.9	66.5
	frame	2	45.1	57.1	61.5	68.7
		3	46.7	58.2	59.3	69.2
		1	45.6	65.9	70.3	78.0
	dict	2	50.5	65.4	69.2	78.6
		3	50.0	67.0	72.5	80.2
		1	49.5	61.5	66.5	78.0
delta	fixed	2	50.5	64.3	70.3	78.0
		3	57.1	66.5	70.9	77.5
		1	47.3	59.3	65.9	76.9
	frame	2	47.3	63.7	69.2	78.6
		3	57.7	68.7	71.4	75.8
		1	42.3	61.5	68.1	78.6
	dict	2	51.1	68.1	72.5	79.1
		3	58.8	67.0	71.4	79.1
		1	49.5	64.3	68.1	78.6
accel	fixed	2	56.0	67.6	70.9	78.0
		3	56.0	64.3	69.8	76.4
		1	50.5	62.6	66.5	76.4
	frame	2	55.5	68.7	69.8	78.0
		3	58.8	67.0	69.8	80.2

Table 7: % Hits per false alarm for the normalised whole-word system with various parameterisations, number of component mixtures, and whole-word model state classes

Param	No. of False Alarms			
	1	2	3	10
static	44.3	56.4	59.2	68.3
delta	47.5	62.2	67.6	77.6
accel	47.4	62.8	67.6	77.9

Table 8: % Hits per false alarm for 1 component mixture models, in the normalised whole-word system, averaged over the *dict*, *fixed* and *frame* state classes

Param	No. of False Alarms			
	1	2	3	10
static	45.3	57.5	61.3	69.4
delta	49.4	64.5	69.6	78.4
accel	54.2	68.1	71.1	78.4

Table 9: % Hits per false alarm for 2 component mixture models, in the normalised whole-word system, averaged over the *dict*, *fixed* and *frame* state classes

Param	No. of False Alarms			
	1	2	3	10
static	44.7	57.5	59.8	70.0
delta	54.9	67.4	71.6	77.8
accel	57.9	66.1	70.3	78.6

Table 10: % Hits per false alarm for 3 component mixture models, in the normalised whole-word system, averaged over the *dict*, *fixed* and *frame* state classes

6.2 Sub-word tests

6.2.1 Effect of parameterisation

Param	No. of False Alarms			
	1 FA	2 FA	3 FA	10 FA
static	45.1	52.2	60.4	69.8
delta	52.2	63.2	67.6	80.2
accel	59.3	67.0	70.9	80.8

Table 11: % Hits per false alarm for (i) static, (ii) plus delta, and (iii) plus acceleration coefficients, 4 component mixture models

Table 11 shows that significant improvements in performance are achieved by adding extra parameters. In particular, a much higher hit rate is achieved at the 1st and 2nd false alarm levels.

6.2.2 Effect of number of components

Param	No. of cpts	No. of False Alarms			
		1 FA	2 FA	3 FA	10 FA
delta	1	45.1	56.0	57.7	67.6
	2	47.8	59.3	59.9	71.4
	3	50.5	61.5	66.5	75.8
	4	52.2	63.2	67.6	80.2
	6	54.4	64.3	72.0	80.2
accel	1	51.6	56.6	59.9	66.5
	2	57.1	65.9	69.2	72.5
	3	58.8	63.7	71.4	79.1
	4	59.3	67.0	70.9	80.8
	6	59.9	69.8	72.0	82.4

Table 12: % Hits per false alarm for 1, 2, 3, 4 and 6 component mixture sub-word keyword models with delta and acceleration parameterisations

Table 12 shows that the hit rate improves as the number of component mixtures is increased.

6.2.3 Performance as a function of keyword syllables

Table 13 shows the average performance for 1, 2 and 3 syllable keywords. In general, performance improves as the number of syllables in a keyword is increased. The improvement is most noticeable at the 1st false alarm level. Wilcox and Bush also noted that increasing the number of syllables in a keyword improved the performance of their sub-word keyword system [9]. They suggested that users could optimise voice editing and indexing systems by using phrases rather than single words.

No. of syllables	No. of False Alarms			
	1 FA	2 FA	3 FA	10 FA
1	47.9	67.3	69.1	86.2
2	63.1	65.2	72.1	75.1
3	76.4	80.6	80.6	80.6

Table 13: % Hits per false alarm for 1, 2, and 3 syllable keywords, using 6 component mixture, acceleration parameter HMMs

6.2.4 Keyword scores

MAIL		ASSESS		WORKSTATION	
Score	T/F	Score	T/F	Score	T/F
103.0	T	103.3	T	102.2	T
103.0	T	102.5	F	102.1	T
102.2	T	102.0	F	102.0	T
101.5	F	101.8	T	102.0	T
101.2	F	101.1	T	100.6	T
101.2	F	100.3	F	100.4	T

Table 14: Putative hit scores and classification, T(RUE HIT) and F(ALSE ALARM), up to the 3rd false alarm and above for the keywords 'find', 'assess', and 'workstation', using 6 component mixture, acceleration parameter HMMs

Table 14 shows examples of the putative hit scores given by the normalised sub-word system. Overall, 10 true hits out of a possible 17 were achieved for the keyword 'mail', 3 true hits out of 3 for 'assess', and 6 true hits out of 6 for 'workstation'. The number of false alarms can be controlled by applying a threshold to the keyword scores. It is not clear what is the most appropriate threshold, for example a simple fixed constant or a keyword dependent variable might be used.

6.3 Comparison of the normalised whole-word and sub-word systems

The best whole-word and sub-word systems achieve a similar hit rate using static and static plus delta coefficients, as shown in table 15. The sub-word system, however, out-performs the whole-word system when acceleration coefficients are added. In particular a large improvement (+8.8%) is seen at the 1st false alarm level. Overall, far more consistent results are observed for the normalised sub-word system than the normalised whole-word system. This is due to the large amount of training data available in the sub-word case, allowing good keyword models to be formed.

Another advantage of the sub-word approach is that it is directly applicable to an open keyword system. This has allowed the same set of monophone HMMs to be used in the VoiceNotes demonstration tool, with a different set of keywords, without requiring any further training of the models.

Keyword model	Param	No. of False Alarms			
		1 FA	2 FA	3 FA	10 FA
Whole-word	static	44.0	56.6	59.9	68.7
	delta	50.5	65.4	69.2	78.6
	accel	51.1	68.1	72.5	79.1
Sub-word	static	43.4	55.5	59.9	74.7
	delta	54.4	64.3	72.0	80.2
	accel	59.9	69.8	72.0	82.4

Table 15: % Hits per false alarm for (i) 2 component mixture, *dict* state class model, normalised whole-word, and (ii) 6 component mixture normalised sub-word systems

7 Future work

The immediate goal of the current research is to develop a VoiceNotes type system to search on arbitrary keywords. In the ideal case, the keyword model is formed based on a single utterance of the keyword [9], i.e. from the user's search request. The baseline approach will be to generate the keyword model by running a Viterbi recogniser using the set of monophone models. The recognised phone string will form the sub-word model of the keyword. This will be placed in parallel with the set of monophone models in the keyword network in the normalised sub-word system. The use of N-best Viterbi recognition to generate the keyword model will also be investigated, and different acoustic units compared (e.g. monophones, triphones, clustered general units [9]).

To date, the availability of a large amount of training data for the speaker has been assumed. Because the target is the general, multi-tasking business professional, it is unrealistic to expect an extensive initial training phase for any device. A product may be expected to have reasonable "walk-up" accuracy on recognition tasks. The word-spotting performance of speaker-independent models, with and without adaptation, will therefore also be investigated. This will include the affect on performance when the user is not the same speaker as the data being searched.

The normalisation rescoring method used in this work to improve the word-spotting performance is not well-founded theoretically. An alternative may be to combine the keyword score with the likelihood ratio methods proposed by Rose [5], [7].

8 Acknowledgements

This work is supported by Hewlett-Packard Laboratories Bristol.

References

[1] 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

[2] Knill, K. M. *Description of Word-Spotting Version of VoiceNotes Demo.* Cambridge University Engineering Dept., Internal Report, unpublished, May, 1994

[3] Lamel, L. F., Kassel, H. K., and Seneff, S. *Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus.* Proc. DARPA Speech Recognition Workshop, pp26-32, 1986

[4] NIST. *The Road Rally Word-Spotting Corpora (RDRALLY1).* NIST Speech Disc 6-1.1, September, 1991

[5] Rose, R. C. and Paul, D. B. *A Hidden Markov Model Based Keyword Recognition System.* Proc ICASSP, S2.24, pp129-132, Albuquerque, April, 1990

[6] Rose R. C. *Techniques for Information Retrieval from Speech Messages.* Lincoln Lab Journal, Vol. 4, No. 1, 1991

[7] Rose R. C. *Discriminant Wordspotting Techniques for Rejecting Non-vocabularly Utterances in Unconstrained Speech.* Proc ICASSP, Vol. 2, ppII-105-II-108, San Francisco, March, 1992

[8] Stifleman, L.J., Arons, B., Schmandt, C., and Hulteen, E.A. *VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker.* Proc. InterCHI, 1993

[9] Wilcox, L.D. and Bush, M.A. *HMM-Based Wordspotting for Voice Editing and Indexing.* Proc Eurospeech, Vol. 1, pp25-28, Genoa, Sept, 1991

[10] 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

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