South African Computer Journal
Number 23
July 1999

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 23
Julie 1999

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

World-Wide Web: `http://www.cs.up.ac.za/sacj/`

Subscriptions

	Annual	Single copy
Southern Africa	R80.00	R40.00
Elsewhere	US$40.00	US$20.00

An additional US$15 per year is charged for airmail outside Southern Africa

to be sent to:

Computer Society of South Africa
Box 1714, Halfway House, 1685
Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

Guest Editorial
Computer Science and Information Systems: The Future?

Philip Machanick

Department of Computer Science, University of the Witwatersrand, South Africa
philip@cs.wits.ac.za

1 Introduction

As president of the South African Institute for Computer Scientists and Information Technologists (SAICSIT), I have visited a number of campuses and companies, in an attempt at arriving at a general assessment of the state of our subjects in South Africa.

An issue which I consistently pick up is that while everyone seems to think that computer-related skills are extremely important and in short supply, our academic departments are also extremely under-resourced.

At the last Southern African Computer Lecturers Association (SACLA) conference (28-29 June, Golden Gate), I had the opportunity to discuss the problems other academics see. This editorial lists some of the problems reported at SACLA, and proposes a way forward.

2 Problems

At SACLA, I led a discussion of problems seen in our academic departments.

There was wide agreement that both Computer Science (CS) and Information Systems (IS) departments were under pressure to increase student numbers (massification), and were seen as cash cows to prop up less popular subjects. It was broadly agreed that staffing was a critical issue: too few posts for the workload, salaries way out of line with industry (half or less, as compared to the US, where an academic salary may be 80% of an industry salary). Recent graduates often make more than professors which makes it hard to persuade our students to become academics (even to do higher degrees). Attracting a recent PhD with a sense of adventure is may be possible, but attracting experienced people used to earning a salary in a strong currency is hard. IS jobs are worse than CS, as the skills required are more like those in business. Support staff salaries are an even harder issue: their skills relate even more directly to job descriptions in industry.

A problem in addressing our concerns is that we are so overworked that we don't have time for "politics": academics with no students have time on their hands, but we don't. More industry support not only with directly addressing problems but with taking on university administrations would be useful, but they too have major problems and don't have free time.

3 Solutions?

Solutions are harder to identify than problems.

The SACLA session ended with a proposal that we conduct surveys of our institutions and businesses, to find out what the problems are, as a starting point for going to university administrations, government and business.

Another idea was to attempt to find common cause with business in taking on problems they have in common with academia, including the skills shortage, the insufficient capacity of our education system, and dealing with employment equity.

One of our biggest difficulties is to free up time to deal with issues such as resource allocation within our universities. The "competition" is frequently other academics with time on their hands, since they have too few students, and therefore are in a position to spend time looking after their interests.

What is needed now is some thought about how to pull ourselves out of the mess we are in. In particular, we need strategies to exploit our strengths: our high demand among students, the high demand for the skills we produce and the ubiquitous applicability of computer technology.

Given the wide use of computers, it would seem obvious that our areas should be strongly supported by a range of role players, yet the fact that so many different groups are interested in computer technology in one way or another has tended to fragment efforts to enhance our industry and academic institutions.

Clearly, from conversations I have held, some departments are in much better shape than others. Even so, some kind of collective effort is likely to achieve more results than if we allow ourselves to be pushed around as individuals. Addressing the fragmentation of efforts seems a worthy goal in itself, to reduce duplication and contradictory goals.

I appeal to anyone who has constructive ideas on how to take our subjects forward to contact me. Let us work on building ourselves up. The economy depends on us, much more than on most other academic disciplines. It's time we made that point, and made it strongly.

SAICSIT'99

South African Institute of Computer Scientists and Information Technologists

Annual Research Conference 17-19 November 1999

Prepare for the New Millennium

Is there life after y2k?

Mount Amanzi Lodge, Hartebeespoort

near Johannesburg and Pretoria

keynote speaker: Barbara Simons, ACM President

and many other local and international speakers (academic and industry)

Call for Participation

papers in a many areas of Computer Science and Information Systems are expected

Price Waterhouse Coopers prizes: Best Paper R10000 • Best Student Paper R5000

please check the conference web site for accepted papers:

http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/

To Register

go to the conference registration web page:

http://www.cs.wits.ac.za/~philip/SAICSIT/SAICSIT-99/reservation.html

or contact **SAICSIT'99 Secretary** *for details:*

Department of Computer Science, Senate House 1137
University of the Witwatersrand
Jorissen Street
Wits, 2050
South Africa

phone (011)716-3309 fax 339-3513 (international: replace 011 by 27-11)

saicsit99-info@cs.wits.ac.za

Dates and Publication Details

early booking deadline: 14 September • on-site registration starts: 17 November 1999

workshops, tutorials 17 November 1999 • paper sessions: 18-19 November 1999

papers will appear in a special issue of South African Computer Journal

sponsors

NRF PRICEWATERHOUSECOOPERS

Connected Digit Recognition in Afrikaans Using Hidden Markov Models

C. Nieuwoudt[a] and E.C. Botha[b]

Department of Electrical and Electronic Engineering, University of Pretoria, Pretoria 0002, South Africa
[a]chris@ee.up.ac.za, [b]botha@ee.up.ac.za

Abstract

We implement and evaluate the performance of a state-of-the-art connected digit recognition system for spoken Afrikaans digits. The creation of a database for our purpose is discussed. Results indicate that an efficient implementation achieving high recognition accuracy (99.2% digit accuracy and 93.3% string accuracy) is attainable. Results also indicate that improvement due to duration modelling is confined to cases where there are relatively few states per hidden Markov model (HMM).

The HMM is used to model the statistical properties of speech signals, as it is well understood and forms a main component of almost all state-of-the-art automatic speech recognition systems. Continuous models, in particular multi-modal Gaussians, are used to model speech feature distributions. Experiments are performed on a database consisting of spoken telephone numbers in Afrikaans from 33 speakers. An HMM is initialised for each digit from labelled isolated digits and embedded re-estimation is used to further train models on continuously spoken telephone numbers. Parametric models, in the form of the Gamma distribution, are used to model word and state duration density. Duration modelling is incorporated as part of the recognition process. Classification experiments are performed on continuous digit test data and performance is measured for a range of modelling parameters.

Keywords: *connected digit recognition, duration modelling*
Computing Review Categories: *G.3, I.5*

1 Introduction

When developing speech recognition systems, the availability of labelled speech data is probably the most important criterion affecting the success of the effort. For most South African languages, including Afrikaans, little speech data is available. As our first endeavour in Afrikaans speech recognition we have therefore decided to develop a task specific speech recognition system which needs only a small vocabulary and does not have the need for complex language modelling. A small sized vocabulary allows us to relatively easily capture and label enough data to train and evaluate a speaker independent recognition system. The absence of complex language modelling implies that the task syntax can be represented in a simple way by a finite state machine.

Connected digit recognition is a prime example of such a task specific system since it has a very small vocabulary and is described by a simple grammar. Previous research on the recognition of Afrikaans digits has focussed on isolated digit recognition[2] and wordspotting[3]. Connected digit recognition is still an actively researched topic [6, 11] and is often used for testing new ideas in automatic speech recognition. Connected digit recognition is useful for many applications such as voice dialling of telephone numbers, automatic data entry, credit card entry, etc.

Rabiner et al[8] achieved very good results using hidden Markov models (HMMs) and cepstral, as well as delta cepstral speech features on the Texas Instruments (TI) connected digits database[5]. Several improvements have been reported on the performance of similar systems also using the TI database by e.g. improving duration modelling[1, 6] or adding filler models[9, 11].

In this paper we report on the creation of an Afrikaans spoken digit database and on results achieved using an HMM system with continuous Gaussian distributions and duration modelling. A digit accuracy of 99.2% and a string accuracy of 93.3% was achieved on unknown length connected digit strings in Afrikaans.

The organisation of this paper is as follows. In Section 2 we discuss the creation of the digit database. In Section 3 we discuss all aspects of the speech recognition system, from preprocessing, feature extraction and training, to the search algorithms which implement the recognition. Section 4 presents the experiments performed and results obtained. We conclude in Section 5.

2 Spoken Digit Database

A graphical user interface on a computer was used in the acquisition of the data in order to automate the process. Once the user was logged into the system after typing in his or her student number, the program gave the user a description of what was expected and proceeded to prompt the user to say various digit strings. The first two digit

number of speakers	33 male
isolated word digit strings	66
continuously spoken digit strings	227
average digit string length	9 digits
sampling rate	8kHz
quantisation	16 bits
average Signal-to-Noise Ratio (SNR)	16.8 dB
lowest SNR for a speaker	5.4 dB
highest SNR for a speaker	23.6 dB

Table 1: Description of Afrikaans digit database

strings were to be spoken as isolated digit strings, i.e. with pauses between digits, and the next ten digit strings were to be spoken continuously, as one would normally say telephone numbers. The digit strings were created at random with a uniform distribution, with the exceptions that each digit appears only once per discretely spoken digit string (to cover all digits equally) and the first three digits of 70% of the continuous strings conform to various South African long-distance and cellular prefixes. String length varies from 7-digit local telephone numbers to 10-digit cellular and long-distance telephone numbers. The reason for imitating real telephone numbers is to have the speakers speak as naturally as possible. It is well known that speech read from prepared text differs from spontaneous speech [12], but people generally each have their own way of saying telephone numbers and the recordings should thus be reasonably natural.

A summary of database parameters is given in Table 1. The database contains speech from male speakers between the ages of 20 and 28. A headset with a built-in microphone was used to record the data. The use of a headset minimises the possibly large gain differences that can be caused by large variations in the distance between the speaker and the microphone[4].

The strings of isolated word digit strings from the first 12 people in the database were labelled at word level. For the rest of the digit strings, including all the continuously spoken digit strings, word level transcriptions, as generated by the acquisition program, were included in the database. The transcriptions were compared to the actual speech and were changed in four cases to compensate for incorrect sequences of spoken digits. In another three cases speech files were missing or were found not to contain digit sequences and were omitted from the database. It is interesting to note at this point that although speaker and recording errors were rare - just more than 1% of recordings - it exceeds the digit error rate achieved with the recognition process.

The speech data was found to be relatively noisy, with a steady hiss of wide band noise appearing in all the recordings. This is most likely due to an impedance mismatch between the sound card and the microphone. In order to estimate the Signal-to-Noise Ratio (SNR), each utterance was Viterbi-aligned with the models corresponding to the word transcription for the utterance. Detail on the training of models and alignment with speech is given in Section 3. Since models were used both for digits and for pauses before, after, and in between speech, the labels could be used to distinguish non-speech from speech. The non-speech segments include not only background noise and noise from the measurement process itself, but also noise generated by the speakers, such as breath intake and release. The SNR was calculated for all the utterances and computed separately for each speaker. A large variation in SNR exists amongst the speakers as is apparent in Table 1. Junqua[4] reports that performance is rather uniform for SNRs greater than 25 dB, but that there is steep degradation in performance as SNRs decrease below this threshold. In our case the average SNR is only 16.8 dB and thus it is expected to limit the achievable performance.

3 Connected Digit Recogniser

The main components used in the training and testing of the speech recognition system[7] we developed are:

- **feature extraction** in which speech signals are converted into sequences of mel-scaled cepstral coefficient vectors along with their time derivatives,
- **training** of HMMs, which includes the Viterbi alignment and Baum-Welch algorithms,
- **pattern matching** in which the feature vectors are matched using dynamic programming to a set of trained HMMs.

We now proceed to discuss each of these items in detail, along with a general description of hidden Markov modelling and duration modelling.

3.1 Feature Extraction

The speech signal is blocked into frames of 16 ms and spaced 10 ms apart - delivering 6 ms of overlap between successive frames. This choice has been empirically determined to deliver good performance. Frames, consisting of 128 samples each, are used to compute Linear Predictive Coefficients (LPCs), from which 13 mel-scaled cepstral coefficients are calculated. The performance was found to be reasonably insensitive to the number of coefficients and we opted for using 13 coefficients, which is also commonly reported in the literature. Temporal information about the speech signal is incorporated by estimating first and second time derivatives for each of the 13 coefficients. A second order linear regression is applied to each set of five consecutive coefficients in order to obtain a smoothed estimate of the first and second time derivatives. The observation sequence **o** used for matching with HMMs thus consists of the 13 mel-scaled cepstral coefficients plus first and second time derivatives, totalling 39 elements, at each frame time. A detailed description of the feature extraction process can be found in [7].

3.2 Hidden Markov Models

An HMM, signified by λ, is described by two sets of parameters

- a state transition matrix $\mathbf{A} = \{a_{ij}\}$ reflecting the probabilities of making transitions from each state i to each other state j,
- a state observation density function $b_j(\mathbf{o})$ reflecting the probability of observing observation vector $\mathbf{o}$ in state j.

The models are first order HMMs since each transition probability to a next state depends only on the current state, and not on which states were previously traversed. Left-to-right HMMs are most commonly used for speech recognition. In left-to-right models the state transition probabilities a_{ij} satisfy the constraints $a_{ij} = 0$ for $i < j-1$ and $i > j$. The assumption is that observation sequences corresponding to the same HMM traverse the same discrete sequence of statistical properties. This agrees with our phonetic understanding of speech as exhibiting piecewise continuous behaviour to a large degree. This unfortunately does not explicitly allow for the modelling of too much variation in the way which the same word may be pronounced other than for time warping of the speech signal.

In our modelling convention, for convenience, we define a dummy initial state number 0 in each HMM. For an HMM with N states, the indices i and j therefore vary from 0 to N. State 0 takes the place of having an initial probability vector for defining allowable transitions to state 1 of the HMM. It will also aid us later in describing the implementation of the level building algorithm used for successive Viterbi searches.

The Markov models are termed "hidden" due to the fact that the states are not observed directly in the observation sequence, but rather indirectly through modelling of observation distributions in each state. Gaussian mixtures are used to model the observation probability density functions. The PDF of observation $\mathbf{o}$ at time t in state j takes the form

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm} N[\mathbf{o}_t, \mu_{jm}, \Sigma_{jm}], \qquad (1)$$

where M is the number of mixture components, c_{jm} is the weight associated with the mth mixture in the jth state, N is the multivariate normal density(not to be confused with the number of states in an HMM), μ_{jm} is the mean vector of the mth mixture in the jth state and Σ_{jm} is the covariance matrix of the mth mixture in the jth state. In order to greatly reduce the number of parameters and since the elements of $\mathbf{o}_t$ are largely uncorrelated, we make the assumption that Σ_{jm} is diagonal. The observation density function expands to

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} c_{jm} \frac{\Pi_{f=1}^{F} e^{-(o_t^f - \mu_{jmf})^2 / 2\sigma_{jmf}^2}}{(2\pi)^{(F/2)} (\Pi_{f=1}^{F} \sigma_{jmf})}, \qquad (2)$$

where o_t^f is the fth element of the observation vector at time t, F is the number of feature vector elements, and σ_{jmf} is the fth covariance on the diagonal of Σ_{jm}.

3.3 Duration Modelling

It is commonly believed that the duration modelling aspect of the HMM approach to speech recognition is a major weakness. Conventional HMMs implicitly model state duration by a Geometric distribution, i.e.

$$p_i(\tau) = a_{ii}^{\tau-1}(1 - a_{ii}), \qquad (3)$$

where a_{ii} is the auto-transition probability in state i and τ is the duration in number of frames. The Geometric distribution is not able to model individual state duration probabilities well since it can only represent an exponentially decreasing probability density function. Explicit duration densities for states may be specified and in such a case the models are called semi-Markov models [7]. State duration density may be modelled with estimated discrete duration probabilities $d_i(\tau), \tau = 1, 2, .., \tau_{max}^i$ for each duration up to a maximum duration τ_{max}^i. This approach has the disadvantage that a large number of parameters has to be estimated. Modelling duration with parametric functions greatly reduces the number of parameters. A popular function for modelling state duration probability is the Gamma distribution

$$p(\tau) = \frac{\alpha^\rho}{\Gamma(\rho)} e^{-\alpha\tau} \tau^{\rho-1}, \qquad (4)$$

which has only the parameters α and ρ that have to be estimated for each state duration model. Initial algorithms for duration modelling were very computationally expensive[7], and a post-processing approach[8] was often used. The post-processing method uses duration metrics to re-score a number of the best paths obtained from a search process. This approach fails where the best re-scored path is not amongst the obtained best paths, and is thus not re-scored. An efficient approach towards incorporating duration modelling into the search process has been proposed by Du Preez[3] and a similar approach was later independently proposed by Burshtein[1]. Both approaches make it possible to add a duration metric at each time frame and to thus obtain the true best path, yet in a computationally efficient manner. We employed the method proposed by Burshtein.

3.4 Hidden Markov Model Training

Separate HMMs are used to model each digit from zero to nine. An HMM is also used to model the pause often found between spoken digits and another HMM is used to model the pause before and after each digit string utterance. Connected word strings are modelled by conceptually stringing together successive HMMs for digits and pauses.

The parameters that have to be estimated in training an N state HMM are:

- $N+1$ independent transition probabilities ($a_{i,i+1} = 1 - a_{ii}$ for the left-to-right model and other off-diagonal values are zero),
- NM mixture weights,
- NMF mean and covariances values and
- $2N$ duration parameters if duration modelling with the Gamma distribution is used.

The parameters are estimated by examining the distribution of features in training data. Initially labelled training samples of each digit are used to train each HMM. In our database, the only labelled samples available are from discretely spoken digits. This training is relatively fast since only a subsection of the total training data is used. Though the discretely spoken digit characteristics probably differ somewhat from those of the continuously spoken digits, it does not matter much since this training serves only to produce models that may converge to good solutions during later training. Training of parameters in each stage is done in batch mode, i.e. parameters are updated after computing statistics from the entire training set for that stage.

The state transition matrix **A** is initialised according to left-to-right constraints. To bootstrap the parameters, each observation feature vector sequence corresponding to a single HMM is subdivided into as many segments of equal length as there are states in the HMM. The means of the Gaussian function in each state are initialised to codebook centroids derived from the corresponding speech feature segments. Each mixture component variance is set to the pooled covariance of the data used to initialise the state. The training is further described in the following sections.

3.5 Viterbi State Alignment

Viterbi state alignment is used to iteratively update the means and covariances of the Gaussian mixture models at each state on labelled data. We give details of the Viterbi dynamic programming algorithm next since it is used both in training and in the recognition process, and we wish to show in some detail the incorporation of duration modelling as well as the level building method for continuous string recognition. We present the Viterbi algorithm, mostly following the syntax from [7]. For an HMM, the Viterbi algorithm finds the best state sequence $\mathbf{q} = (q_1 q_2 .. q_T)$ for a given observation sequence $\mathbf{O} = (\mathbf{o}_1 \mathbf{o}_2 .. \mathbf{o}_T)$, as well as the probability associated with this sequence. We define

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 .. q_{t-1}, q_t = i, \mathbf{o}_1 \mathbf{o}_2 .. \mathbf{o}_T | \lambda], \quad (5)$$

as the highest probability along a single path, at time t, which accounts for the first t observations and ends in state i. By induction, the Viterbi recursion is defined as

$$\delta_{t+1}(j) = \max_{0 \le i \le N} [\delta_t(i) a_{ij}] b_j(\mathbf{o}_{t+1}), 1 \le j \le N. \quad (6)$$

The probability in the final state at the final time frame, $\delta_T(N)$, indicates the score for the match between model and observation sequence. When the Viterbi algorithm is used to align speech with model states, such as is used in training, it is necessary to keep track of the path followed via

$$\psi_{t+1}(j) = arg \max_{0 \le i \le N} [\delta_t(i) a_{ij}]. \quad (7)$$

This path can be backtracked from $\psi_T(N)$ to deliver the highest scoring path.

Note that in Equation 6 we have included the possibility of transitions from state 0 to the current state. In order to initialise the Viterbi search we define $\delta_0(0) = 1$, $\delta_0(i) = 0, i \neq 0$ and $\delta_t(0) = 0, t > 0$. When we discuss the implementation of successive Viterbi searches in a later section, take note that $\delta_t(0)$ will be starting points that are likely not zero.

The actual training of parameters takes place after statistics from an entire batch of training utterances are collected. The result of each application of the Viterbi algorithm is a state-aligned set of observation features. For each feature vector it is determined which mixture in the corresponding state is closest to it. The feature vector is then used to collect mean and variance statistics for that mixture. After statistics have been collected for the batch of training samples, new mean and variance values are computed for the Gaussian mixture models. This process is repeated iteratively until either convergence occurs, or a predetermined number of iterations have been completed.

3.6 Expectation Maximisation

The Baum-Welch method (also known as the expectation maximisation or EM method) is used to perform final training of the HMMs. It is first applied on the labelled training data. The EM method iteratively updates the means, covariances, mixture weights and state transition probabilities at each state. The forward-backward algorithm is used to obtain statistics from the training utterances. Full detail regarding use of the forward-backward algorithm and subsequent parameter update using the EM method can be found in [7].

In order to use EM training on unlabelled training data, each succession of HMM models matching a transcription of a training utterance is conceptually connected together to form a long sequence of states. This connected string of states is used for training in the normal way using the forward-backward algorithm. Since no explicit word boundary information is present, the training allows the boundaries of the HMMs freedom and contextual information is trained into the models.

Lastly, training of durational parameters are done through the Viterbi state alignment of connected HMMs to the utterances they represent. For each alignment, the sum of the first and second moments of the number of frames corresponding to each state in each HMM is collected. The empirical expectation values of the mean($\hat{E}\{\tau\}$) and

variance($\hat{E}^2\{\tau\}$) of each duration can be calculated and used to obtain the Gamma distribution parameters ($\hat{\alpha}$ and $\hat{\rho}$) through

$$\hat{\alpha} = \frac{\hat{E}\{\tau\}}{\widehat{VAR}\{\tau\}}, \hat{\rho} = \frac{\hat{E}^2\{\tau\}}{\widehat{VAR}\{\tau\}}. \quad (8)$$

This concludes the training process. In the next section we discuss the implementation of the recogniser.

3.7 Pattern Matching

We firstly discuss the incorporation of duration modelling into the Viterbi algorithm. Duration modelling is implemented according to the synchronous frame by frame method suggested by Burshtein[1]. The method modifies the Viterbi recursion (Equation 6) by incorporating a duration penalty $C^t_{i,j}$ of making a transition from state i to state j at time $t+1$ within the term that is maximised by the recursion. When written in log format for implementation efficiency, the maximisation term in the Viterbi recursion becomes

$$\max_{0 \le i \le N} [log(\delta_t(i)) + log(a_{ij}) + log(C^t_{i,j})]. \quad (9)$$

To compute the duration penalty, the method keeps track of the number of successive self-transitions in each state. The duration $D_i(t)$ of a state i at time t is equal to one plus the number of successive self-transitions in that state. Let M_i denote the duration at which the Gamma distribution at state i reaches a maximum value, and let $l(u) = log(p(u))$, where $p(u)$ is the Gamma distribution. The duration penalty $C^t_{i,j}$ is then given by

$$C^t_{i,j} = \begin{cases} 0 & i = j, D_i(t) < M_i \\ l(D_i(t+1)) - l(D_i(t)) & i = j, D_i(t) \ge M_i \\ l(D_i(t)) & i \ne j, D_i(t) < M_i \\ l(M_i) & i \ne j, D_i(t) \ge M_i. \end{cases}$$

The working of the method can be understood in the following way. The duration probability density function is used to modify the probability of a transition occurring, based on the duration spent in the state from which the transition occurs. When a transition to a different state is taken, the exact duration is known and can be used to modify the probability. In considering self-transitions, however, the penalty can not be incorporated on a frame by frame basis since the eventual duration in a state is yet unknown. Incorporating the duration probability at each frame as if it were the last time step in a state would penalise initial self-transitions in a state – causing an incorrect bias towards transitions from the previous state. Therefore the method should not penalise self-transitions until the peak duration probability is reached in a state. After the point of peak duration probability, duration penalty is applied in accordance with duration probability density.

With duration modelling now incorporated into the Viterbi search, we turn our attention to the implementation of the level building algorithm. When recognition of a sequence of spoken words is attempted, it is desired to find the best match across all possible sequences of digit and pause models. An exhaustive search of depth R, containing V possibilities in each level leads to V^R Viterbi alignments or in our case at least 10^{10} Viterbi alignments (if pause models are ignored) for a string of 10 digits or less - which is not computationally feasible. The level building algorithm[7] dramatically reduces the computational cost by performing only V searches at each of the R levels, thus effectively $V \times R$ Viterbi alignments.

The level building algorithm works by computing at each successive level l the most likely final state probability (P^l_t) at each frame t over all V models in the search path

$$P^l_t = \max_{1 \le k \le V} [\delta^k_t(N)]. \quad (10)$$

After a level has been completed, the final state probabilities are used as initial state probabilities for Viterbi searches at the next level, i.e. we now set $\delta_t(0) = P^l_t$. This process continues until the desired number of levels have been searched. The most likely sequence ends at the level given by

$$\arg\max_{1 \le l \le R} (P^l_T). \quad (11)$$

From the most likely final state at the final frame it is easy to backtrack the complete path followed through all levels provided that the backtracking information from each individual Viterbi alignment has been retained. Note that the most likely solution does not necessarily present itself at the last level. The level building technique can thus be used to find unknown length word strings up to the maximum depth for which was searched.

4 Results

The data were separated into a training set comprising discretely and continuously spoken digits from 12 speakers, and a test set comprising only continuously spoken digits from the remaining 21 speakers. All experiments are for a speaker independent scenario since there is no overlap between speakers in the test and training sets. The training set was used as described in Section 3 in training the models. Discretely spoken digits were used to initialise the models and the continuously spoken data was used for further training.

The models that were trained include the ten Afrikaans digits from zero to nine, a three state HMM for the pause before and after entire utterances and a one state HMM to model the optional pauses between spoken digits. The one state HMM is the only model which does not strictly follow the left-to-right convention for transitions. It allows a skipping transition so that the model may either be followed or bypassed, whichever is more likely at any level of the search. The grammar requires the first and last model to be the three state pause model, with in between them a series of digits with optional one state pauses between each consecutive pair of digits. The problem we wish to

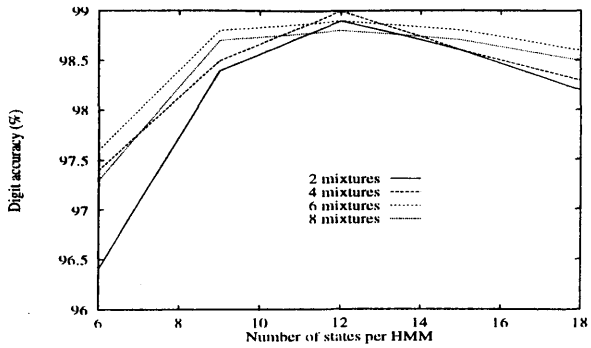

Figure 1: Classification performance on connected digit experiments measuring digit accuracy without explicit duration modelling

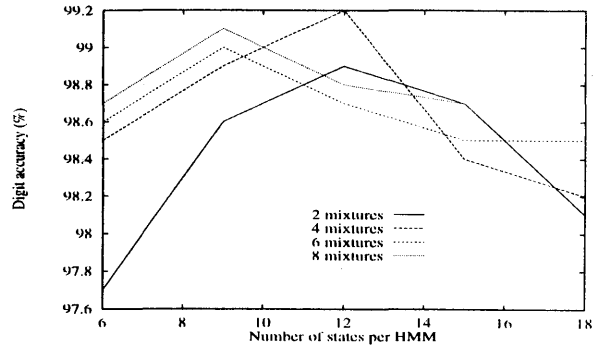

Figure 2: Classification performance on connected digit experiments measuring digit accuracy, including duration modelling

address is the recognition of unknown length digit strings and therefore the maximum search depth was chosen to exceed the longest test string length. The longest test strings contain 10 digits. A search constraint of 12 digits or less was found to be adequate and recognition output on the test set never exceeded 11 digits with properly trained models.

The level building algorithm, as described in Section 3, was used for classification of the test strings. The output of this algorithm is a sequence of digits, and has to be matched against the transcription of each test string. A dynamic programming matching algorithm is used to perform the matching since there is not necessarily a one-to-one matching and digit insertions and deletions may occur in the recognition process. The results are measured in terms of the percentage of correct digits and in terms of digit accuracy. The number of correct digits in a string is simply the highest number of digits that match both value and sequence in the reference string. Digit accuracy is computed to reflect the effect of digit insertion and deletion by the search algorithm and is defined by the total number of digits, minus insertions, deletions and substitutions. Digit accuracy is thus a better indicator of recognition performance than using the percentage of correct digits alone.

The results in Fig. 1 show the digit accuracy achieved as a function of N (with parameter M) without using duration modelling on the unknown length connected digit strings. Peak digit accuracy of 99.0% is achieved with 12 state, 4 mixture HMMs. The results show that accuracy initially increases as more states are used in the HMMs, and then declines as the number of states increase beyond 12. This result is to be expected since, as the number of states increase, the HMM is able to express a more accurate model. When the number of states becomes large, two factors may cause performance to decrease: Firstly the minimum length of the HMM model may approach the length of some of the shorter digits - causing the deletion rate to increase rapidly, and secondly the large amount of parameters leads to a higher variance estimate. The trade-off between bias and variance can also be observed w.r.t. the number of Gaussian mixtures used. Performance initially increases as more mixtures are used, especially when few states are also used. As the number of mixtures grows larger than 4, performance starts to decrease again.

Fig. 2 shows the digit accuracy achieved when state duration modelling is used in the recogniser. Performance generally improves over the case without duration modelling and peak digit accuracy increases from 99.0% to 99.2% (a 20% reduction in error rate), using 6 state, 4 mixture HMMs. Performance increases are most apparent where each HMM contains few states. This is attributed to duration modelling effecting a large reduction in digit insertions. Over all experiments involving 6 state HMMs, the average accuracy increases from 97.2% to 98.4% (a 43% reduction in error rate) by adding state duration modelling. At the same time the percentage digits correct only increases with 0.2%. Most of the improvement is due to a large decrease in the number of digit insertions. When using 12 state HMMs, duration modelling does not change the average accuracy of 98.9% achieved without duration modelling. When HMMs with more than 12 states are used, duration modelling actually degrades performance.

A peak string accuracy of 93.3% is achieved with 6 state, 4 mixture HMMs. This is very close to a predicted 93.0% string accuracy for strings of independent digits, given the mixture of string lengths (30% are 7 digit and 70% are 10 digit strings) and the digit accuracy of 99.2%. Most erroneous strings have only a single digit error and it seems that digit errors are independent of each other.

The results are very similar to those reported in literature on spoken English digits. Rabiner et al[8] achieved a digit string error rate of 2.94% on strings of 1 to 7 digits for speaker independent tests. Under the assumption that digit errors are independent and that strings lengths are uniformly distributed, this translates to a 0.75% digit error rate as compared to the 0.8% digit error rate that we achieved. A technique which was found to greatly improve their performance was the training of multiple HMMs for each digit by clustering the training data for each digit into from 1 to 6 different groups. This technique is usually applied to words for which multiple likely pronunciations exist, often due to dialectic differences. Rabiner et al[8] report almost halving the error rate through application of this technique. Unfortunately this technique assumes that a large quantity of training data is available and could thus not be tried on our database.

Rahim et al[9] report digit string recognition perfor-

mance of 96.1% and 90.3% on two databases of telephone speech. The data includes non-vocabulary words and the system was able to reject 99.9% of non-vocabulary strings while rejecting 5% of correct strings.

Zeljkovic et al[11] report reducing word error rates on connected digit telephone speech from 2.5% to 0.9% by expanding digit and silence models (filler models) from the context independent case to context dependent models. Each word model is substituted by three models, a left context, centre context and right context model. Left and right context models are trained based on the preceding and following word respectively.

5 Conclusion

We have detailed the compilation of a database of spoken digits in Afrikaans, together with the development of a recognition system for continuous speech through application of hidden Markov modelling techniques. The performance achieved (99.2% digit accuracy and 93.3% string accuracy) is comparable to that of leading systems for spoken English digits, even though the SNR of our database is low and a limited amount of training data is available.

Our experiments indicate an interesting trend w.r.t. the relationship between the number of states and the duration modelling capabilities of hidden Markov models. Duration modelling has been reported to substantially improve performance of HMMs for connected digit recognition, using 8 states per HMM[1] (43% string error rate reduction) in one study and using between 5 and 10 states per HMM in another[8] (improvement not quantified). We also find that the improvement in performance due to duration modelling is substantial, but only when there are fewer than the optimal number of states per HMM and that this improvement decreases as the number of states increases. Wang[10] reasoned that the simple structure of linear HMMs is potentially capable of modelling phone duration quite well, but could not demonstrate this i.t.o. recognition results. Our results indicate that HMMs are capable of modelling duration well when enough states are present to deliver optimal performance. For practical systems a tradeoff may exist between the computational cost of adding duration modelling and that of having more states per HMM.

6 Acknowledgements

The authors would like to thank the Center for Spoken Language Understanding at the Oregon Graduate Institute of Science & Technology for their CSLU Toolkit, which was used for feature extraction and part of the training.

References

[1] D. Burshtein. Robust parametric modeling of durations in hidden Markov models. In *Proc. ICASSP '95*, pages 548 – 551, Detroit, MI, May 1995.

[2] J.A. Du Preez. Approaches to speaker independent isolated word recognition. In *COMSIG-88 Proceedings. 1988 IEEE South African symposium on communications and signal processing*, pages 41–45, University of Pretoria, Aug. 1988. IEEE.

[3] J.A. Du Preez. Modelling durations in hidden markov models with application to word spotting. In *COMSIG-91 Proceedings. 1991 IEEE South African symposium on communications and signal processing*, pages 1–5, Fourways, Aug. 1991. IEEE.

[4] Jean-Claude Junqua. Impact of the unknown communication channel on automatic speech recognition: A review. In *Proc. Eurospeech '97*, pages KN–29–KN–32, Rhodes, Greece, September 1997.

[5] R. G. Leonard. A database for speaker-independant digit recognition. In *Proc. ICASSP '84*, pages 42.11.1–42.11.4, 1984.

[6] Miroslav Novak. Improvement on connected digits recognition using duration constraints in the asynchronous decoding scheme. In *Proc. Eurospeech '97*, pages 159–162, Rhodes, Greece, September 1997.

[7] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[8] L. R. Rabiner, J. G. Wilpon, and F. K. Soong. High performance connected digit recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-37(8):1214, 1989.

[9] M. G. Rahim, C.-H. Lee, and B.-H. Juang. Robust utterance verification for connected digits recognition. In *Proc. ICASSP '95*, pages 285 – 288, Detroit, MI, May 1995.

[10] X. Wang. *Incorporating Knowledge on Segmental Duration in HMM-Based Continuous Speech Recognition*. PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, April 1997.

[11] Ilija Zeljkovic and Shrikanth Narayanan. Novel filler acoustic models for connected digit recognition. In *Proc. Eurospeech '97*, pages 283–286, Rhodes, Greece, September 1997.

[12] Victor Zue. Conversational interfaces: Advances and challenges. In *Proc. Eurospeech '97*, pages KN–9–KN–18, Rhodes, Greece, September 1997.

Received: 19/5/98, Accepted: 19/10/98

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications of Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript
Manuscripts for *review* should be prepared according to the following guidelines:

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
- the title (as brief as possible)
- the author's initials and surname
- the author's affiliation and address
- an abstract of less than 200 words
- an appropriate keyword list
- a list of relevant Computing Review Categories
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text according to the Harvard. References should also be according to the Harvard method.

Manuscripts accepted for publication should comply with guidelines as set out on the SACJ web page,

`http://www.cs.up.ac.za/sacj`

which gives a number of examples.

SACJ is produced using the LaTeX document preparation system, in particular LaTeX 2_ε. Previous versions were produced using a style file for a much older version of LaTeX, which is no longer supported. Please see the web site for further information on how to produce manuscripts which have been accepted for publication.

Authors of accepted publications will be required to sign a copyright transfer form.

Charges
Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30.00 per final page for contributions which require no further attention. The maximum is R120.00, prices inclusive of VAT.

These charges may be waived upon request of the author and the discretion of the editor.

Proofs
Proofs of accepted papers may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Letters and Communications
Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words. Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book Reviews
Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement
Placement of advertisements at R1000.00 per full page per issue and R500.00 per half page per issue will be considered. These charges exclude specialised production costs, which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 23, July 1999
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 23, Julie 1999
ISSN 1015-7999

Contents