Start-Synchronous Search for Large Vocabulary Continuous Speech Recognition

Steve Renals[*] and Mike Hochberg[†]

(Preprint of paper to appear in IEEE Trans. Speech and Audio Processing)

Abstract

In this paper, we present a novel, efficient search strategy for large vocabulary continuous speech recognition. The search algorithm, based on a stack decoder framework, utilizes phone-level posterior probability estimates (produced by a connectionist/HMM acoustic model) as a basis for *phone deactivation pruning* — a highly efficient method of reducing the required computation. The single-pass algorithm is naturally factored into the time-asynchronous processing of the word sequence and the time-synchronous processing of the HMM state sequence. This enables the search to be decoupled from the language model while still maintaining the computational benefits of time-synchronous processing. The incorporation of the language model in the search is discussed and computationally cheap approximations to the full language model are introduced. Experiments were performed on the North American Business News task using a 60,000 word vocabulary and a trigram language model. Results indicate that the computational cost of the search may be reduced by more than a factor of 40 with a relative search error of less than 2% using the techniques discussed in the paper.

Keywords — large vocabulary continuous speech recognition, stack decoding, search, hidden markov model, phone deactivation pruning

1 INTRODUCTION

The search problem in large vocabulary continuous speech recognition (LVCSR) can be simply stated: find the most probable sequence of words given a sequence of acoustic observations, an acoustic model and a language model. This is a demanding problem since word boundary information is not available in continuous speech and each word in the dictionary may be hypothesized to start at each

[*] Steve Renals is with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, UK. (email: s.renals@dcs.shef.ac.uk)

[†] Mike Hochberg is with Nuance Communications, Menlo Park, CA 94025, USA. (email: mmh@nuance.com)

frame of acoustic data. The problem is further complicated by the vocabulary size (typically 20,000 words or larger) and the structure imposed on the search space by the language model.

When formulated in a statistical manner, the goal of the speech recognizer is to find the word sequence $\hat{W} = w_1 w_2 \dots w_N$ with the maximum *a posteriori* (MAP) probability given the sequence of acoustic observations $\mathbf{X} = \mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_T$:

$$\hat{W} = \operatorname*{argmax}_{W} P(W|\mathbf{X}) \ . \tag{1}$$

Here, $P(W|\mathbf{X})$ is the *a posteriori* probability of word sequence W given acoustic data $\mathbf{X}$ and can be expressed (using Bayes' Rule) as a product of the *acoustic model* likelihood $p(\mathbf{X}|W)$ and the *language model* prior probability $P(W)$[1]:

$$\begin{aligned} P(W|\mathbf{X}) &= \frac{p(\mathbf{X}|W)P(W)}{p(\mathbf{X})} && (2) \\ &\propto p(\mathbf{X}|W)P(W) \ . && (3) \end{aligned}$$

Note that $p(\mathbf{X})$ may be neglected during recognition (selection of the optimal word sequence) since it is independent of the word sequence.

The acoustic models generally used in statistically-based speech recognition systems are hidden Markov models (HMMs) [1, 2]. The language model is typically an $(n-1)$th order Markov chain on the word sequence and is often referred to as an *n-gram* model. Using HMMs, a hierarchical model of speech may be constructed where an utterance model (a model of a word sequence) is composed of a series of concatenated word models which, in turn, are composed of subword models. The topology and transition probabilities of the utterance model are defined by the language model, the pronunciation dictionary and the topologies of the basic subword HMMs.

By applying (3), marginalizing over all possible state sequences Q and accounting for the structure of the HMM, we may express the MAP recognition criterion (1) as:

$$\begin{aligned} \hat{W} &= \operatorname*{argmax}_{W} \sum_{Q} P(\mathbf{X}|Q,W)P(Q|W)P(W) && (4) \\ &= \operatorname*{argmax}_{W} P(W) \sum_{Q} P(Q|W)p(\mathbf{X}|Q) \ . && (5) \end{aligned}$$

If the *Viterbi* criterion is employed, this summation is approximated using the most probable state sequence:

$$\begin{aligned} \hat{W} &\simeq \operatorname*{argmax}_{W} P(W) \max_{Q} P(Q|W)p(\mathbf{X}|Q) && (6) \\ &= \operatorname*{argmax}_{W} P(W) \max_{Q \in \mathcal{Q}_W} P(Q)p(\mathbf{X}|Q) && (7) \end{aligned}$$

[1] As usual, P and p denote the probability mass and density functions, respectively. Precise notation requires conditioning these statistical quantities on the acoustic and language models. For clarity, however, we drop the explicit dependence.

where $\mathcal{Q}_W$ is the set of HMM state sequences $Q = q_1 q_2 \dots q_T$ that correspond to the word sequence W: $\mathcal{Q}_W = \{Q : P(Q, W) > 0\}$. Although (7) does not directly optimize the word-recognition criterion specified in (1), it has been shown to work well in practice.

The task for the search algorithm is to evaluate (5) or (7) — ie, determine $\hat{W}$ given the various models and the acoustic data. Direct evaluation of all the possible word sequences is impossible (given the large vocabulary) and an efficient search algorithm will consider only a very small subset of all possible utterance models. Typically, the effective size of the search space is reduced through pruning of unlikely hypotheses and/or the elimination of repeated computations. Two basic classes of search procedure have been used for continuous speech recognition: time-synchronous Viterbi decoding and time-asynchronous stack decoding.

1.1 Viterbi Decoding

In its basic form, Viterbi decoding (or forward dynamic programming) may be regarded as an efficient, recursive algorithm that performs an optimal exhaustive search [3]. For HMM-based speech recognition, the Viterbi algorithm is used to find the most probable path through a probabilistically scored time/state lattice[2]. Viterbi decoding is efficient for small problems with a finite state language model and is guaranteed to find the optimal path. However, the computational expense of an exhaustive dynamic programming search is too great for LVCSR problems. Various pruning and data organization techniques have been adopted to reduce the effective size of the search space. One such algorithm — beam search [6, 7] — defines a pruning beam width Δ relative to the most probable hypothesis log likelihood $\log P_{max}(t)$ at frame t. Hypotheses outside the beam (those with log likelihoods less than $\log P_{max}(t) - \Delta$) are pruned from the search.

The use of time-synchronous Viterbi decoding becomes more complex when long-span language models (eg trigrams) are introduced. This requires multiple copies of word models to account for different contexts and can be extremely resource-intensive for large vocabulary tasks. Approaches taken to address this problem include the use of dynamic search structures [8] and multi-pass search where more simple, early passes direct the later passes [9].

1.2 Stack Decoding

The ideas underpinning stack decoding are those of sequential decoding in communications theory [10] and of heuristic search in artificial intelligence, such as the A^* algorithm [11]. These search algorithms are time asynchronous — the best scoring path or hypothesis, irrespective of time, is chosen for extension and this process is continued until a complete hypothesis is determined. The crucial function for these algorithms is the estimated score (log likelihood in this case)

[2] This approach was first used in speech recognition by Vintsyuk [4] and a tutorial by Ortmanns et al can be found in [5].

of hypothesis h at time t, $f_h(t)$, and is given by

$$f_h(t) = a_h(t) + b^*_h(t) \ . \tag{8}$$

Here, $a_h(t)$ is the score of the partial hypothesis using information to time t and $b^*_h(t)$ is the estimate of the best possible score (maximum log likelihood) in extending the partial hypothesis to a valid complete hypothesis. It has been shown that as long as $b^*_h(t)$ is an upper bound on the actual log likelihood, then the search algorithm is admissible [11] (ie, no errors will be introduced that would not occur if an exhaustive search was performed).

Stack decoding is a best-first algorithm. The *best* (partial) hypothesis (ie, the hypothesis for which $f_h(t)$ is greatest) from the "stack" of hypotheses under consideration is popped from the stack, extended by a word, and the extended hypotheses pushed back onto the stack. Provided the estimate of $f_h(t)$ is admissible, the first complete hypothesis to be popped from the stack will correspond to the most probable utterance model. Such algorithms have been investigated by Bahl et al [12, 1, 13, 14], Paul [15], Kenny et al [16] and Soong and Huang [17]. In the statistical framework for speech recognition, (8) may be expressed as:

$$\begin{aligned} f_h(t) &= \log p(W_h^{(1,t)}, \mathbf{X}^{(1,t)}) + \\ &\quad \log \sum_{W^{(t+1,T)}} p(W^{(t+1,T)}, \mathbf{X}^{(t+1,T)} | W_h^{(1,t)}, \mathbf{X}^{(1,t)}) \ . \end{aligned} \tag{9}$$

The first term on the right hand side, $\log p(W_h^{(1,t)}, \mathbf{X}^{(1,t)})$, corresponds to $a_h(t)$ in (8) and is the joint log likelihood of the sequence of acoustic vectors to time t and the word sequence hypothesized to that time. The second term, $\log \sum_{W^{(t+1,T)}} p(W^{(t+1,T)}, \mathbf{X}^{(t+1,T)} | W_h^{(1,t)}, \mathbf{X}^{(1,t)})$, corresponds to $b^*_h(t)$ in (8) and — in this case — is the sum over all possible word sequences that can follow $W_h^{(1,t)}$ while accounting for the remaining acoustic evidence $\mathbf{X}^{(t+1,T)}$. Computing this quantity exactly is out of the question in practice! Various approximations have been employed. Bahl et al [18, 1] ignored the dependence on $W_h^{(1,t)}$ and estimated the expected value of $\sum_{W^{(t+1,T)}} p(W^{(t+1,T)}, \mathbf{X}^{(t+1,T)} | \mathbf{X}^{(1,t)}) = p(\mathbf{X}^{(t+1,T)} | \mathbf{X}^{(1,t)})$ using a first-order Markov model computed from the acoustic training data. Kenny et al [16] and Soong and Huang [17] used a multi-pass framework where an initial Viterbi pass with simple acoustic and language models was used to approximate $b^*_h(t)$ for an A^* search proceeding in the opposite direction.

The disadvantage of the above approaches to approximating $b^*_h(t)$ is the requirement to look ahead at the acoustic data. An alternative approach [13, 14, 15] — and the approach presented in this paper — does not rely on looking ahead. Instead, $b^*_h(t)$ is constructed such that hypotheses with earlier reference times always have higher scores than those with later reference times. This method is discussed in greater detail in section 2.1.

Stack decoding has several potential advantages over Viterbi decoding:

- The language model is decoupled from the acoustic model and is not used to generate new recognition hypotheses;

- It is easy to incorporate non-Markovian knowledge sources (eg long-span LMs) without massively expanding the state space;

- The Viterbi assumption is not embedded in the search and thus a full maximum likelihood search criterion may be used with little or no computational overhead.

Disadvantages of the approach include sensitivity to the choice of heuristic (a bad choice of $b_h^*(t)$ can lead to an explosion in the effective search space) and the possibility of repeated computation.

1.3 Search with Abbot

In this paper, we present a search procedure developed for the Abbot LVCSR system[3] [19, 20]. Abbot is a connectionist/HMM system [21] and the search procedure takes advantage of some features particular to this approach (although the search algorithm described in this paper is applicable to standard HMM systems as well). This approach differs from traditional HMMs in that the posterior probability of each phone given the acoustic data is directly estimated at each frame, rather than the likelihood of a phone (or state) model generating the data. This posterior probability estimation is achieved by using a connectionist network trained as a phone classifier. In the Abbot system, a recurrent network [22] is used for the acoustic model (figure 1). Direct estimation of the posterior probability distribution using a connectionist network is attractive since fewer parameters are required for the connectionist model (the posterior distribution is typically less complex than the likelihood) and connectionist architectures make very few assumptions on the form of the distribution. Additionally, this approach allows pruning to be based on the posterior probability estimates (see section 3).

Since the likelihood is required in the decoding process, the posterior is converted to a scaled likelihood, $L(\mathbf{x}; q)$. This is computed by dividing the posterior probability estimate of phone (or HMM state) q given the data $\mathbf{x}$ by the class prior $P(q)$:

$$L(\mathbf{x}; q) = \frac{P(q|\mathbf{x})}{P(q)} = \frac{p(\mathbf{x}|q)}{p(\mathbf{x})} \ . \tag{10}$$

$P(q)$ is estimated as the relative frequency of q in the training data. The assumptions underlying this acoustic model are discussed in detail in [21, 23].

[3] A demonstration version of the system, AbbotDemo, is available from: ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/recognition/AbbotDemo/

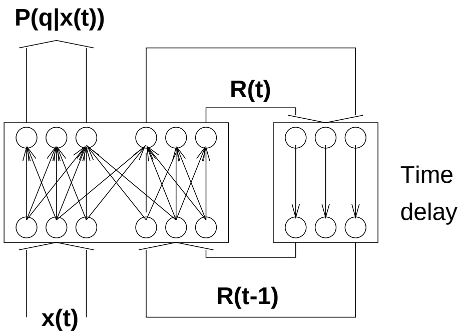


Figure 1: Recurrent network architecture used for acoustic modelling in the connectionist/HMM approach. The acoustic vector $\mathbf{x}(t)$ is input to the trained network, which outputs posterior phone probability estimates $P(q|\mathbf{x}(t))$. $R(t)$ represents the internal state vector of the recurrent network.

2 SEARCH ORGANIZATION AND ALGORITHM

2.1 Start-synchronous Search

The single-pass search algorithm presented here utilizes a *start-synchronous*[4] strategy. This strategy factors the search into a time-asynchronous processing of the word sequence and a time-synchronous processing of the HMM state sequence. A benefit of this approach is that it enables the search to be cleanly decoupled from the language model (LM), while maintaining the computational benefits of time-synchronous processing.

A *hypothesis* h is specified by a reference time t_h, a word string $W_h = w_h(1)w_h(2)\ldots w_h(n_h)$ and a score LP_h. LP_h is the log probability of the word string W_h describing the acoustics from time 1 to time t_h, given the acoustic and language models[5]. The ABBOT system is based on direct posterior probability estimation, and thus LP_h is an estimate of the log posterior probability $\log P(W_h|\mathbf{X}^{(1,t_h)})$; in a "standard" HMM system LP_h will correspond to an estimate of the joint density $\log p(W_h, \mathbf{X}^{(1,t_h)})$. The search algorithm applies in either case. LP_h may be estimated by summing over all state sequences as in (5) or by using the Viterbi approximation and considering only the most probable state sequence as in (7). A *stack* of hypotheses is a priority queue data structure [24], typically implemented as a binary or Fibonacci heap, which efficiently supports the required operations of popping the most probable hypothesis and inserting new hypotheses.

In start-synchronous search, hypotheses are processed in increasing order of reference time t_h; this is equivalent to storing hypotheses in a sequence of stacks,

[4] This term was suggested to us by John Bridle (personal communication).

[5] Since the time range is unambiguous, the word sequence notation has been simplified from $W_h^{(1,t)}$ to W_h.

one for each reference time. This start-synchronous, multi-stack implementation is used here and the basic decoding algorithm is outlined in figure 2. The array of stacks, *stacks*, is processed sequentially with the earliest reference time first. The stack of hypotheses to be processed is extended by a single word using the acoustic model. As extended hypotheses are generated with reference time $t' > t_h$, they are inserted into the stack of hypotheses with reference time t' (figure 3). Hypotheses with the same reference time are processed in parallel, allowing different LM contexts to share the same acoustic model computations.

```
DECODE( ){
    $start\_t := 0$
    **while** $start\_t < utterance\_length$ {
        $st := stacks[start\_t]$
        $st$.PRUNE($LUB[start\_t], \Delta$)
        $lexicon$.EXTEND($st, stacks, t$)
        $start\_t := start\_t + 1$
    }
}
```

Figure 2: The basic start-synchronous stack decoding algorithm. Stacks are processed in reference time order, each hypothesis on the stack is checked for validity (inside the pruning beamwidth) and the surviving hypotheses are extended by one word using the tree structured pronunciation models (*lexicon*).

The acoustic model consists of a set of HMMs, one for each pronunciation of each word. These word models are defined by a pronunciation dictionary and constructed by concatenating the relevant subword HMMs. In this work, the subword HMMs are context-independent phone models[6], although context-dependent phone models have been used successfully with this search algorithm [26]. For computational efficiency, the pronunciation models are represented using a tree structure (figure 4) in which each node corresponds to an instantiation of a basic subword HMM. Tree structuring [27, 28, 29] allows pronunciations with a similar prefix to share memory and computation when being evaluated. The root node of the tree corresponds to a single state silence model which may be skipped; this allows possible pauses between words to be modelled. Word ends do not necessarily correspond to leaves of the tree (nodes without successors): all leaves must correspond to a word end, but internal nodes of the tree can also correspond to word ends (as well as being internal nodes of other words).

A hypothesis h is extended by one word w using such a pronunciation tree, resulting in an extended hypothesis $h' = h.w$ with reference time $t_{h'}$. This

[6] Empirical speech recognition results [25] indicate that the connectionist/HMM approach offers much richer context-independent phone models compared with Gaussian mixture models.

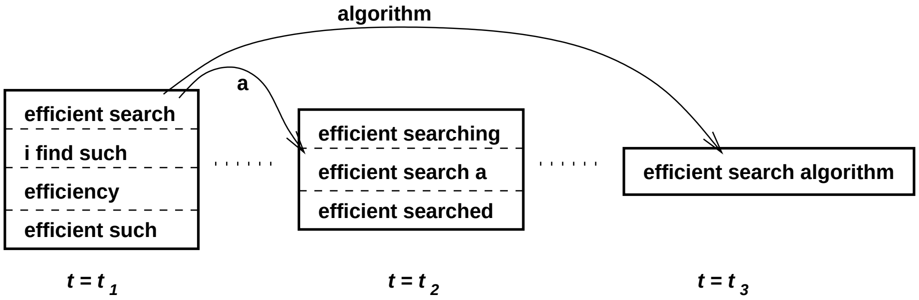


Figure 3: Illustration of the start synchronous search strategy. The stack of hypotheses with reference time $t = t_1$ is being processed. The most probable hypothesis at this time ("efficient search") is extended by the most probable one word extensions ("a" and "algorithm" are illustrated). The resultant extended hypotheses are inserted into the stack at their reference time — in this case "algorithm" has duration $t_3 - t_1$. In practice all hypotheses with identical reference times are extended in parallel.

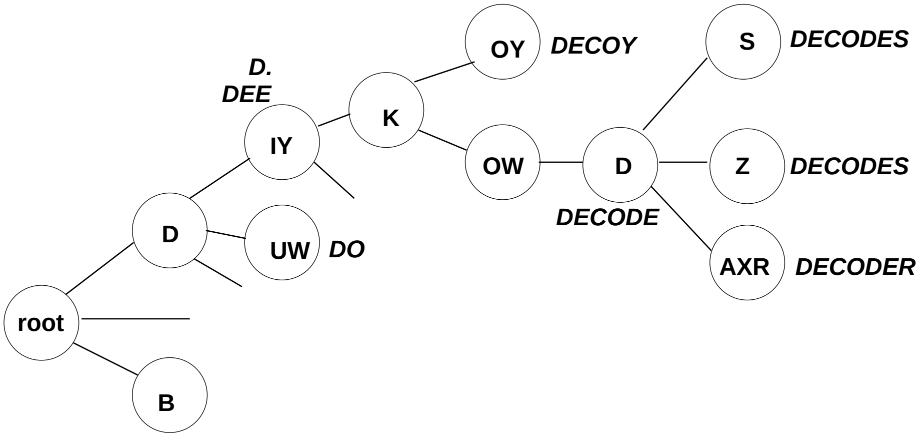


Figure 4: A fragment of a pronunciation prefix tree. The root node corresponds to a single state pause model that may be skipped.

```
(1)   lexicon.EXTEND(st, stacks, start_t){
(2)       t := start_t
(3)       DEPTH_FIRST_UPDATE_LUB()
(4)       active_list[t].CLEAR()
(5)       root_node.ACTIVATE()
(6)       active_list[t].PUSH(root_node)
(7)       while t < utterance_length
          and active_list[t].SIZE() > 0 {
(8)           while active_list[t].SIZE() > 0 {
(9)               node := active_list[t].POP()
(10)              node.FORWARD()
(11)              node.ACTIVATE_SUCCESSORS(active_list[t])
(12)              if node.PRUNE(LUB[t + 1], $\Delta$)
(13)                  node.DEACTIVATE()
(14)              else
(15)                  active_list[t + 1].PUSH(node)
(16)              if node.exit_state.ACTIVE()
                  and node.IS_WORD_END()
(17)                  node.EXTEND_HYPS(st, t, stacks[t])
(18)          }
(19)          t := t + 1
(20)      }
(21)  }
```

Figure 5: The algorithm for extending a set of hypotheses by one word using the tree-structured set of pronunciation models. The *active_list*$[t]$ contains the set of nodes activated at time t and the algorithm continues until *active_list*$[t']$, $t' > t$, is empty (or the end of the utterance is reached).

procedure is outlined in figure 5. Following (10), the scaled likelihood for this extension may be written:

$$L(\mathbf{X}^{(t_h,t_{h'})}; w') = L(\mathbf{X}^{(t_h,t_{h'})}; Q^{(t_h,t_{h'})}) \tag{11}$$

$$= \prod_{i=t_h}^{t_{h'}} L(\mathbf{x}(i); q(i)) P(q(i)|q(i-1)) \tag{12}$$

where $Q^{(t_h,t_{h'})} = q(t_h)\ldots q(t_{h'})$ is the optimal state sequence (assuming the Viterbi criterion) for w' given $\mathbf{X}^{(t_h,t_{h'})}$. (The full scaled likelihood may be similarly defined using the forward probabilities in place of the optimal state sequence.) To compute $L(\mathbf{X}^{(t_h,t_{h'})}; w')$, the root node of the tree is initially activated (line 5 of figure 5). In this case, the possible set of extension words includes the complete vocabulary. The pronunciation tree search is carried out in a breadth-first (time-synchronous) manner. When the exit state of a word-end node (corresponding to word w') is activated at time $t_{h'} > t_h$, then an extended hypothesis $h' = h.w'$ with reference time $t_{h'}$ is created and pushed onto the stack at time $t_{h'}$ with the corresponding log probability $LP_{h'}$ (line 17):

$$LP_{h'} = LP_h + \log L(\mathbf{X}^{(t_h, t_{h'})}; w') + \log P(w'|W_h) \quad . \tag{13}$$

The LM log probability $\log P(w'|W_h)$ is accessed when a word extension is hypothesized. Candidate words are proposed solely by their acoustics. This process continues until all possible extended hypotheses have been created and pushed onto the relevant stack. Before adding a hypothesis to the stack, a check is performed to determine if any of the hypotheses currently on the stack are "LM equivalent" (ie, they have the same LM state). If there is an LM equivalent hypothesis, then the one with lower probability is discarded (Viterbi criterion) or the two hypotheses are merged by summing probabilities (forward decoding criterion). For efficiency, all unpruned hypotheses in a stack with reference time t_h are popped and extended in parallel. If cross-word context-dependent phone models are not used, then the probability of extending a hypothesis h by a word w only depends on h via the LM and may be incorporated at the word end (see section 2.3). In the case of cross-word models, additional book-keeping and delayed model evaluation is required [30].

This algorithm is easily extended to generate word graphs (lattices) without incurring extra computation. The elements of a word graph consist of the hypothesized word extensions with the corresponding start and stop times, acoustic log probabilities and LM probabilities. These terms can all be found in (13).

2.2 Pruning

As already alluded to in figures 2 and 5, the above process will be extremely inefficient without an effective pruning algorithm such as beam search. Beam search requires the computation of a reference score referred to as the *least upper bound* on the log probability of the most likely hypothesis. An accurate estimate of lub$[LP(t_h)]$ (the least upper bound) is essential if this heuristic is to lead to an efficient search. We have investigated two approaches to estimating lub$[LP(t_h)]$:

Greedy strategy: whenever a partial extension to a hypothesis has log probability:

$$[LP_h + L(\mathbf{X}^{(t_h,t)}; Q^{(t_h,t)})] > \text{lub}[LP(t)]$$

then update:

$$\text{lub}[LP(t)] := [LP_h + L(\mathbf{X}^{(t_h,t)}; Q^{(t_h,t)})].$$

Backtrace strategy: update lub$[LP(t)]$ using paths obtained by backtracing from complete word extensions. If $LP_h^{bt}(t)$ is the log probability at time t of the most probable path for hypothesis h and:

$$LP_h^{bt}(t) > \text{lub}[LP(t)]$$

then update:

$$\text{lub}[LP(t)] := LP_h^{bt}(t).$$

The greedy strategy requires no additional computation, but it is not optimal. In particular, states that do not result in a complete, unpruned word extension may contribute to the update of $\text{lub}[LP(t)]$ and the constraints of the LM on the single word extension are not applied. The backtrace strategy involves a little more computational resources, but it ensures that $\text{lub}[LP(t_h)]$ is updated using only paths corresponding to complete word extensions. A slightly more elaborate version of this idea is used in the envelope search strategy of Gopalakrishan et al [14]. A comparison of the backtrace and greedy estimates of $\text{lub}[LP(t_h)]$ for a typical sentence is shown in figure 6, together with the log probability trace of the most probable complete hypothesis.

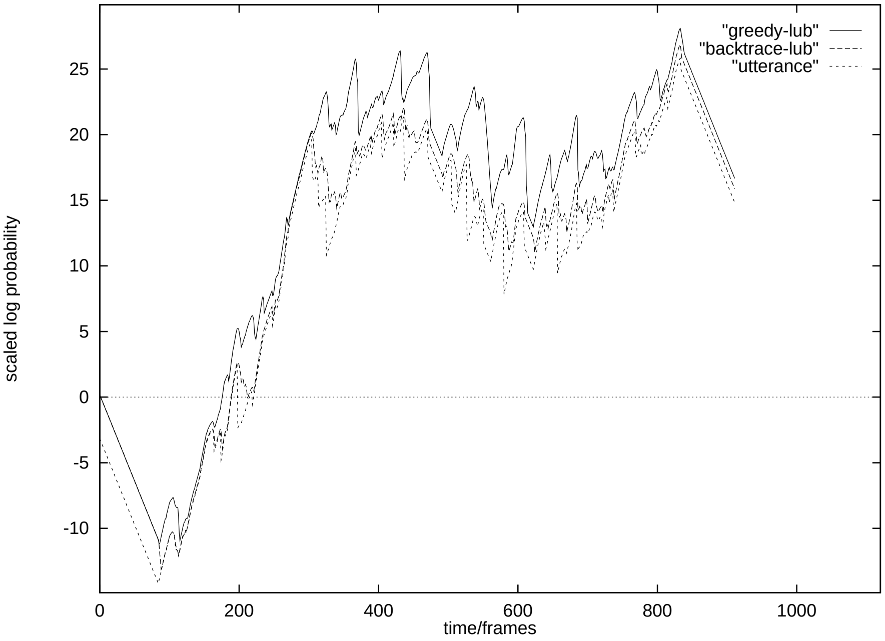


Figure 6: Least upper bound estimates produced by the search algorithm using the greedy and backtrace strategies. The upper curve is produced by the greedy strategy and the middle one by the backtrace strategy. The lower curve corresponds to the probability trace of the most probable complete hypothesis.

A garbage model of any random sequence of words or speech sounds is used to initialize $\text{lub}[LP(t_h)]$. This can either be a separate garbage model evaluated for each frame or an estimate obtained from the local phone posterior probability estimates in a similar fashion to the “online garbage” approach in [31]. In the online garbage approach – which was adopted for this work – the n most probable phone posteriors (excluding the most probable) are averaged and converted to a scaled likelihood by dividing by a uniform prior. This average likelihood is combined with a nominal Markov process score.

The initial estimate of $\mathrm{lub}[LP(t_h)]$ is further refined at each start time (when a set of hypotheses is to be extended by a word) using a simple depth-first process (line 3 of figure 5). This considers a single path down the tree by always considering the most probable successor node at any point, until all successor nodes are outside the beam. The initial estimate of $\mathrm{lub}[LP(t_h)]$ is then updated by backtracing from any complete word extensions found on this path. This process is an efficient way to improve the conservative garbage model initialization of the least upper bound, while requiring few extra models to be evaluated.

Having arrived at an estimate of the least upper bound on the log probability, beam search pruning may be used with $\mathrm{lub}[LP(t)]$ as a reference for a beam with width Δ. Hypotheses, or partially extended hypotheses, with a log probability $LP_h < (\mathrm{lub}[LP(t_h)] - \Delta)$ are pruned from the search. The beam is used in pruning at three points in the search: when activating nodes in the pronunciation tree (lines 12–15 in figure 5); when pushing extended hypotheses on to the relevant stack (line 17 in figure 5); and when choosing hypotheses to extend (line 5 in figure 2).

State-level pruning of nodes in the pronunciation tree is the most critical point of pruning. Let $L(\mathbf{X}^{(t_h,t)}; i)$ denote the scaled likelihood of the acoustic data $\mathbf{X}^{(t_h,t)}$ being generated by a node sequence $root, a, \ldots, i$ (where this is the most probable state sequence through the pronunciation tree resulting in state i at time t). If:

$$LP_h + \log L(\mathbf{X}^{(t_h,t)}; i) < \mathrm{lub}[LP(t)] - \Delta \tag{14}$$

then node i is pruned from the search. $L(\mathbf{X}^{(t_h,t)}; i)$ is the extension log probability of the most probable state of node i given the acoustic data, so a node is pruned only if all its states fall outside the pruning beamwidth. Note that a node may be reactivated if its parent (or any of its ancestors) remain activated.

The pruning beam is also applied at word ends when pushing an extended hypothesis onto a stack and when preparing to extend a hypothesis. In both these cases, if

$$LP_h < \mathrm{lub}[LP(t_h)] - \Delta \ , \tag{15}$$

then hypothesis h is pruned (not pushed onto the stack, or not considered for extension). Pruning prior to extension is required since a hypothesis that was not pruned when it was pushed on to a stack may no longer be within the beam when it is to be extended due to $\mathrm{lub}[LP(t_h)]$ being updated by other hypotheses under consideration.

State-level pruning takes place using the log probability of the most probable hypothesis being extended. Hypotheses are differentiated at word ends where the difference in log probability between hypotheses and the LM score are considered. Thus state-level pruning affects all hypotheses being propagated in parallel, whereas word end pruning (when pushing extended hypotheses on to appropriate stacks) selects between hypotheses being extended.

Specifying a maximum size s for each stack provides a second level of pruning. If a hypothesis h falls within the beam (ie $LP_h \geq \text{lub}[LP(t_h)] - \Delta$), then it is only added to the appropriate stack if the stack has less than s entries or if $LP_h > \min_{h' \in \text{stack}} LP_{h'}$. In the case where the stack has s entries, adding a new entry deletes the least probable hypothesis from the stack. This maximum stack size criterion has the same effect as adaptively decreasing the beamwidth Δ in areas of the search with many competing hypotheses.

2.3 Language model

In the basic search organization described above, the LM is applied at word ends only and with all candidate extensions to hypotheses being generated solely by the acoustic model. This allows an extremely simple interface between the LM and the search via a function that returns the probability $P(w|w_h(1), w_h(2), \ldots, w_h(n))$ of a hypothesis h being extended by a word w. An important benefit of this relationship between the search and the LM is the possibility of decoding using a larger acoustic model vocabulary than LM vocabulary — not every word represented in the pronunciation tree need be represented explicitly in the LM, provided it contains probabilities for “unknown” words. This means that new words can be added to the recognition system by providing a pronunciation from which an acoustic model may be constructed, but without the requirement to recompute the LM.

Pruning of the search could be tightened further if the LM could be applied before word ends. However, early application of the LM is made more complex by both the tree structure (each node may be part of the pronunciation model for several words) and the parallel extension of hypotheses (several different LM contexts need to be considered). This also has the effect of coupling the search to the LM.

The simplest way to incorporate LM information into the search is to utilize LM information which can be precomputed. One approach is to use a unigram approximation to the LM, by computing $P(w)$ for all words w. At each node, the maximum unigram probability of all words containing this node in a pronunciation is computed. During the search, this unigram upper bound at a node is incorporated as a context-independent estimate of extending any hypothesis through that node. This method has been referred to as *unigram smearing* [32]. A second approach involves computing an upper bound on the LM probability given the context — specifically, if a hypothesis being extend has final word w_j then $P^*(w_j) = \max_{w_i, w_k} P(w_j|w_i, w_k)$ (in the case of a trigram) is used to approximate the LM probability, with the maximum being computed over all possible extension words and the remainder of the LM context. These approximations may be computed in advance and stored in a table. We refer to this method as the *context upper bound* approximation.

Exact LM probabilities can be used in the search, but at some computational expense. Such information may be incorporated by smearing the LM probabilities through the tree: the LM probability for a node given a hypothesis (context) is the maximum LM probability over all words that pass through that node. The computational expense involved depends on the LM lookup, although this may be decreased through caching LM probabilities that have been recently accessed (or are expected to be accessed). An intermediate position between incorporating exact LM probability upper bounds at each node and incorporating exact LM probabilities at word ends only is to incorporate probabilities at some nodes and to use the statically computed approximations for the rest. A reasonable parameter to control which nodes should use exact LM probabilities is based on the number of pronunciations passing through a node. Nodes for which no more than a certain number (referred to as the "LM incorporation threshold") of pronunciations pass, incorporate the exact LM probability. The reasoning behind this approach is that the upper bound is only likely to be significantly better than the static approximations when it is computed by taking a maximum over relatively few words.

The parallel propagation of hypotheses impacts smearing of exact LM probabilities through the tree, since each node requires a LM upper bound for each LM context (each hypothesis). If this information is computed, then it may be most efficient to prune hypotheses individually at each node according to the hypothesis tree entry probability and the exact LM upper bound based on the LM context of each hypothesis. This is referred to as "state level hypothesis pruning". Experiments using this and the other LM incorporation strategies are reported in section 4.

If exact LM probabilities are not incorporated within words, then within-word log probabilities will be greater than word-end log probabilities. This can be taken advantage crudely (but effectively) by using a smaller pruning beamwidth at the state level compared with the word level.

3 PHONE DEACTIVATION PRUNING

In addition to the likelihood-based pruning described above, we have also developed a posterior probability-based pruning strategy that takes advantage of the connectionist/HMM approach. This pruning strategy exploits the fact that the connectionist acoustic model estimates posterior probabilities rather than likelihoods. (Posteriors may be regarded as discriminative probabilities that do not incorporate an estimate of $P(\mathbf{x})$). Direct estimation of the phone posteriors avoids the need to sum over baseform HMMs, which would be required to carry out an equivalent approach in a likelihood-based system — a costly computation for a large, context-dependent system.

For each frame of data, the connectionist acoustic model produces a complete vector of context-independent posterior phone probabilities. These phone posteriors may be regarded as a local estimate of the presence of a phone at a particular time frame. If the posterior probability estimate of a phone given a frame of acoustic data is below a threshold, then all words containing that phone at that time frame may be pruned (deactivated), ie

$$\textbf{if}\ \ P(q_i|\mathbf{x})\ <\ \text{threshold}\ \ \textbf{then}\ \ P(q_i|\mathbf{x})\ :=\ 0$$

We refer to this process as *phone deactivation pruning* [33]. The posterior probability threshold used to make the pruning decision may be empirically determined using a development set and is constant for all phones. The effect of varying this threshold on both recognition accuracy and CPU time is reported in section 4.

Phone deactivation pruning is related to the channel-bank-based approach of Gopalakrishnan et al. [34], which uses likelihoods to carry out a similar pruning operation. However, the channel-bank approach is somewhat more complex and requires phone-dependent thresholds. Reported experimental results for the channel-bank approach indicated that it offered a factor of two speedup with a 5–10% increase in relative search error.

4 EXPERIMENTS

Experiments have been carried out with the start-synchronous search algorithm using the DARPA North American Business News (NAB) task. This large vocabulary, speaker-independent, continuous speech recognition task uses read speech data. The experiments reported here were performed using the Abbot system trained on the WSJ0 short-term speaker training corpus (SI–84), consisting of around 15 hours of speech from 84 speakers. The test set used in these experiments was the DARPA/NIST Hub 3 Evaluation Test Set (1995)/Sennheiser microphone (C0 condition), consisting of roughly 45 minutes of speech (300 utterances) recorded from 15 speakers. Parameters and search strategies were not developed on this evaluation set: other NAB development data sets were used for this purpose. A 60,000 word vocabulary was used with the standard (1995) back-off trigram language model for this task, containing 8.5 million trigram probabilities and 7.4 million bigram probabilities. The Abbot acoustic model consisted of two context-independent recurrent networks with 53 context-independent phone classes (plus silence). One network estimated the phone posterior probability distribution for each frame given a sequence of 12th order perceptual linear prediction features. The other network performed the same distribution estimation with features presented in reverse order[7], and the two probability estimates were averaged in the log domain.

The criteria that we have used to evaluate the search algorithm are:

Word Error Rate: The transcription word error rate obtained by summing the deletion, insertion and substitution errors.

Search Error: The error due to the search algorithm pruning the most probable complete hypothesis at some intermediate point. It is quantified as

[7] Note that recurrent networks maintain contextual information via the state units and are time-asymmetric.

the increase in word error rate relative to the unpruned case. Note that an increase in pruning can cause a reduction in word error rate due to "lucky" search errors in which a hypothesis has a fewer number of substitution, deletion or insertion errors compared to one with higher probability.

Active Model Count: The mean number of phone model evaluations per frame (ie the mean number of nodes in the pronunciation tree evaluated per frame). Phone models may be evaluated multiple times at each frame corresponding to extensions from different start times.

Hypothesis Creation Count: The mean number of extended hypotheses created each frame.

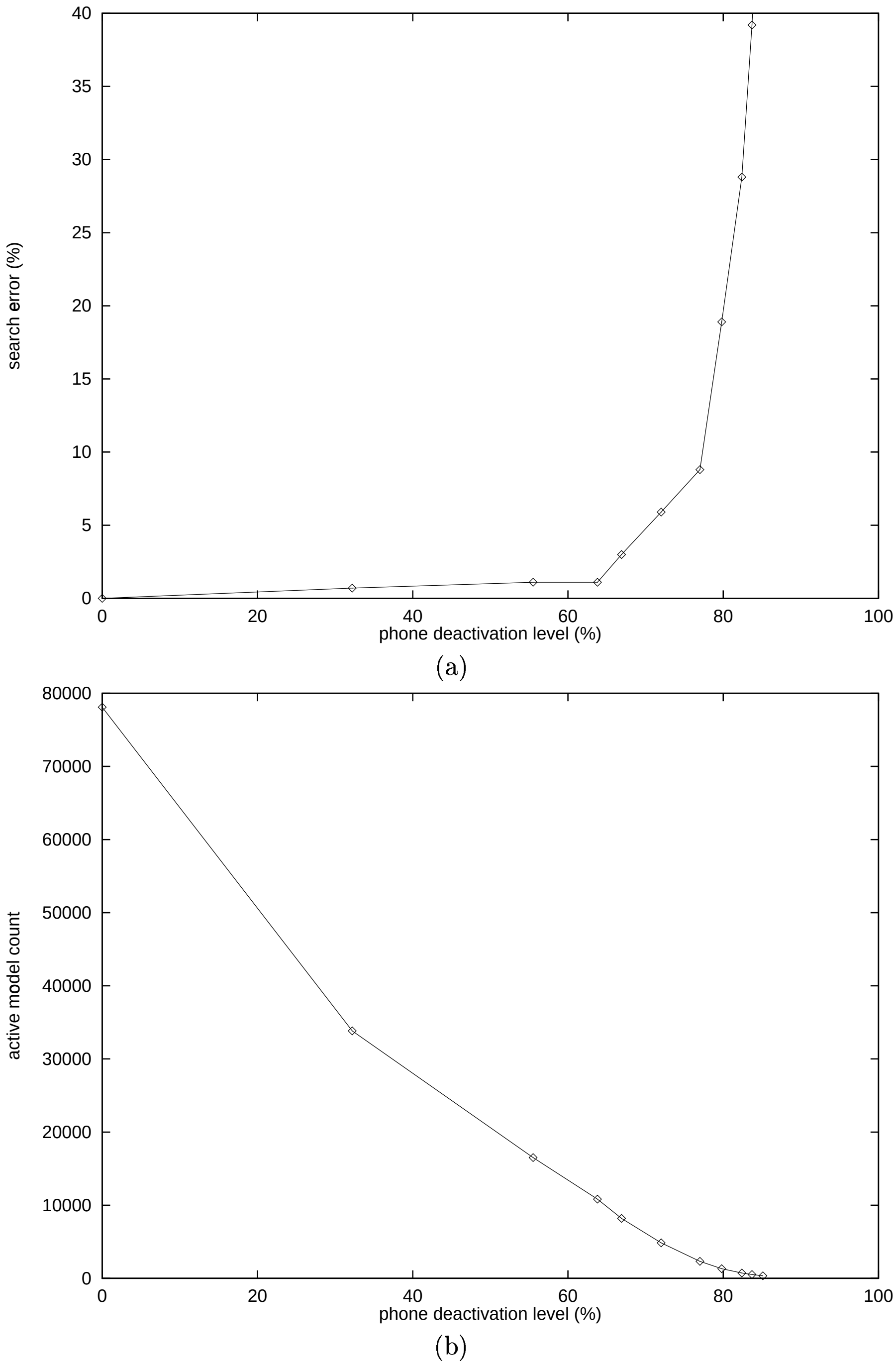
Phone Deactivation Level: The effectiveness of phone deactivation pruning can be expressed by the percentage of phone models that are pruned at each time frame, averaged over all time frames. The raw percentage will tend to be an over-estimate of the computational saving of this pruning method, since it is not weighted by the prior probabilities of phones (ie pruning frequently occurring phones leads to bigger computational savings than pruning infrequent phones). Therefore, we report the phone deactivation level as the percentage of phones pruned weighted by phone prior probability (estimated using the relative frequencies in the training data).

We also express the performance in terms of the running time of the software implementation (on a Sun Ultra 1/167) — all figures refer to the search time and do not include the computation time needed to perform the feature extraction and run the recurrent networks (which is around 1.25 × realtime on this machine). All results are averaged over the test set of 300 utterances.

The experiments on the phone deactivation and beam pruning (sections 4.1 and 4.2, respectively) were carried out using a maximum stack size of 31 hypotheses per reference time (determined empirically on development test data). These experiments did not use exact incorporation of the LM within words, but some LM information was incorporated using the context upper bound method (section 2.3). Section 4.3 reports on experiments in which LM probabilities were incorporated in the search at the state level.

4.1 Phone Deactivation Pruning

The first set of experiments investigated the effectiveness of the posterior probability based phone deactivation pruning. A wide beamwidth was chosen so that the base case in which phone deactivation pruning was not applied could be regarded as having no search errors. In this case, the overall word error was 15.3% and all search errors are computed relative to this value. Figure 7 shows the search error and the active model count plotted against the phone deactivation level. At the phone level, an operating curve for the phone deactivation pruning process may be obtained by plotting the phone deactivation level against the "correct phone" deactivation level. This measures the percentage of "correct" phones (obtained by a Viterbi alignment) that are pruned (figure 8).



(a)

(b)

Figure 7: Variation of (a) search error and (b) active model count with phone deactivation level. A phone deactivation level of 0.0 corresponds to no pruning (threshold = 0), a phone deactivation level of 64% corresponds to a threshold of 70.0×10^{-6} and a phone deactivation level of 82% corresponds to a pruning threshold of 2000.0×10^{-6}. The search used a beam of 6.0 and a "state beam" of 5.0, together with the backtrace LUB estimation.

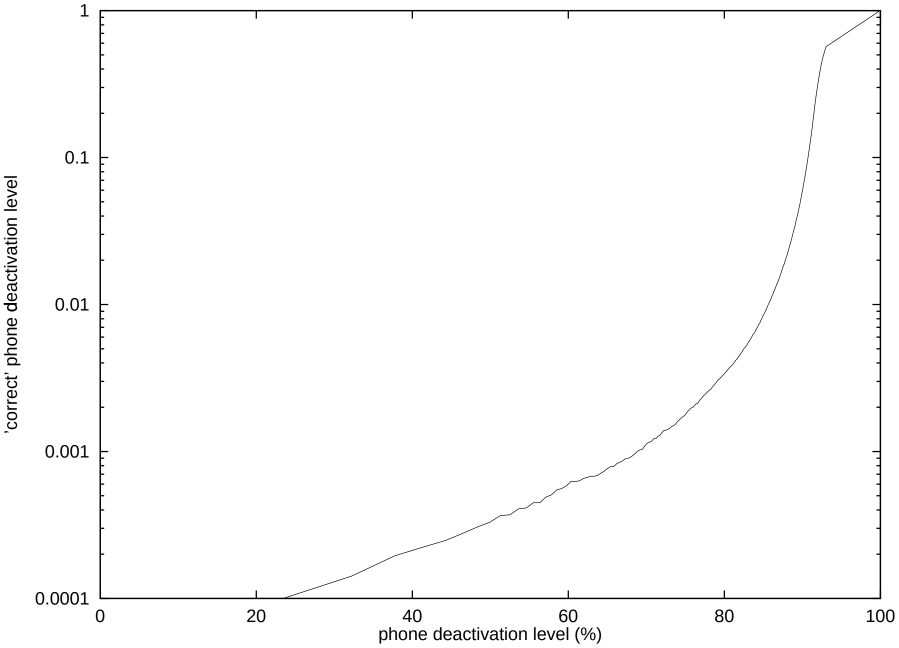


Figure 8: Plot of the "correct phone" deactivation pruning level versus the phone deactivation pruning level.

These results indicate that a substantial amount of phone deactivation pruning may be applied without a significant impact on search accuracy. In particular, the average number of phone model evaluations may be reduced by a factor of 7 with around 1% search error and by a factor of 30 or more with around 8% search error. In this case, the running time scales almost linearly with the active model count: on an UltraSparc 1/167, the decoding ran in 90× realtime if phone deactivation pruning was not applied (at this setting of the beam); with a phone deactivation level of 64%, the run time was reduced to 15× realtime (with a relative search error of 1%).

The trade-off between search error and computation is relatively independent of speaker. As in all search algorithms, a mismatch between training and testing acoustics (e.g., high noise level, etc.) results in less differentiation between competing hypotheses causing a slower process. The relative improvements in the speed, however, remain the same. Figure 9 shows the search error plotted against the phone deactivation level for five of the fifteen speakers in the test set.

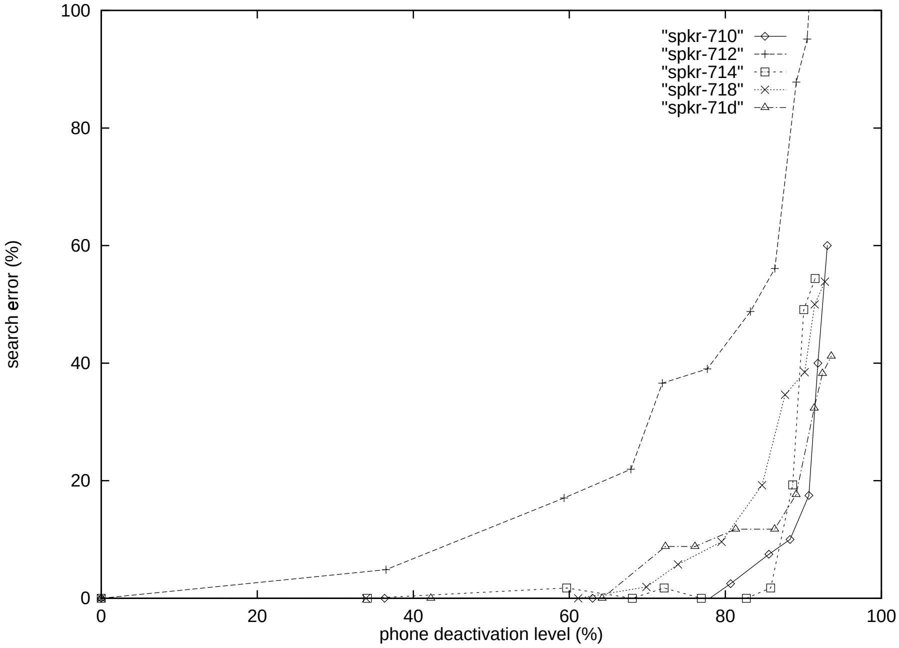


Figure 9: Plot of the search error vs. phone deactivation level per speaker, for five of the fifteen speakers in the test set (20 utterances per speaker). Note that the relationship between the phone deactivation level and the posterior probability threshold is speaker-dependent.

4.2 Beam Search and LUB estimation

The next set of experiments investigated the influence of the beam search on the behaviour of the algorithm. Since the least upper bound contains LM information covering the current frame (at least in the case of the backtrace strategy), more efficient pruning can be achieved using a narrower beam within words (as compared with word ends) when only minimal LM information is incorporated within words. Figure 10 illustrates how the search error and two measures of search effort (active model count and hypothesis creation count) vary with changing beamwidths. This figure indicates that the active model count is largely dependent on the state-level beamwidth and independent of word-level beamwidth, while the opposite is true for the hypothesis creation count. The active model count and the hypothesis creation count are the most CPU intensive portions of the algorithm and the overall running time depends linearly on both counts.

The results in figure 10 also show the difference between the greedy and backtrace strategies for setting the LUB estimate. When the pruning beamwidth is larger, there is little to choose between the two strategies — indeed, the greedy strategy may be preferable since it does not have the memory and computational expense of storing backtrace information. However, the backtrace strategy is more stable to reductions in the state-level beam relative to the word-level beam and has vastly better performance if small search errors may be tolerated. For example, if a search error of less than 5% is acceptable, then the best parameter setting for the greedy strategy results in around 5000 active models/frame (approximately 5× realtime). The backtrace strategy, however, can achieve less than 5% search error with an active model count of roughly 2300 (less than 2× realtime). If realtime processing is desired (at this level of phone deactivation pruning), then the backtrace strategy achieves this with under 10% search error, whereas the performance of the greedy strategy explodes to over 30% search error.

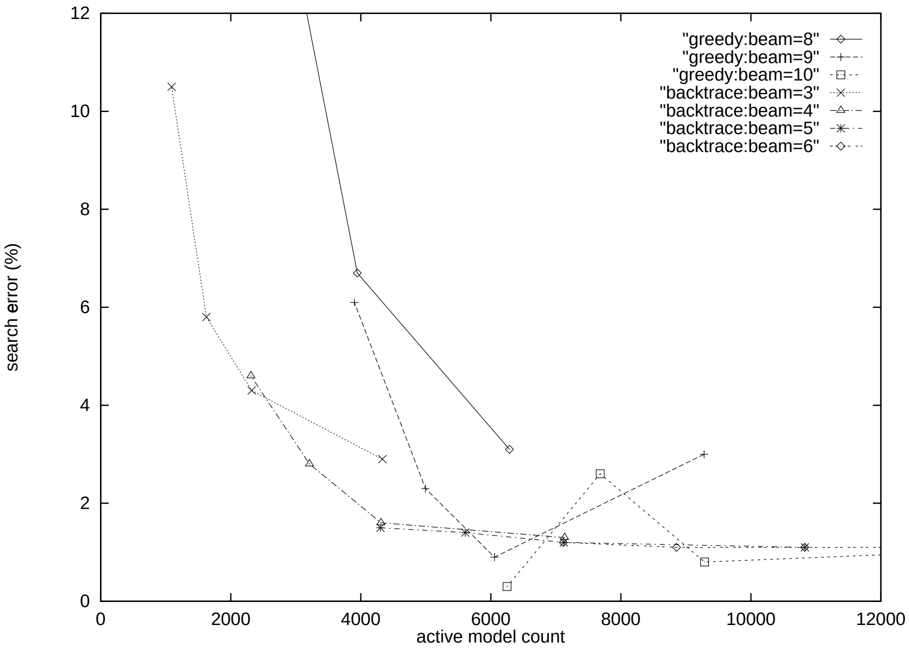
4.3 Language Model Incorporation

The previous experiments did not incorporate LM information at the state-level, except through the context upper bound method described in section 2.3. In this section we report on a set of experiments in which LM information is incorporated at the state level. There is a trade-off between the accuracy of the state level LM information and the computation required to incorporate the LM at the state level. The degrees of LM incorporation we consider are:

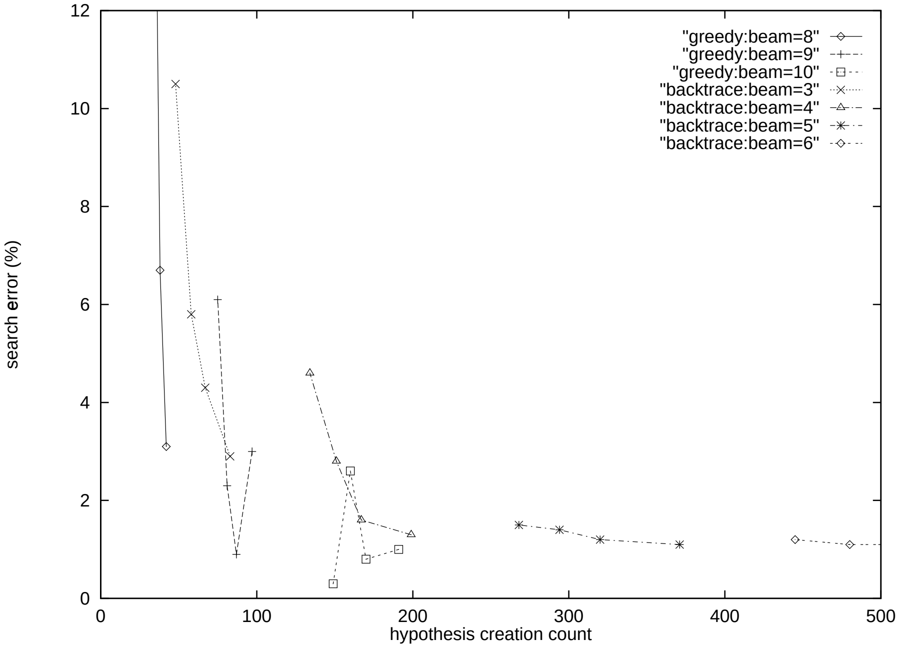
1. No LM incorporation;

2. Context upper-bound approach — upper bound on LM probability is computed by maximizing over possible long-term current words and long-term contexts given the preceding word;

3. Unigram LM approximation at the state level (unigram smearing) — LM



(a)



(b)

Figure 10: Plot of search error vs. (a) active model count and (b) hypothesis creation count for the greedy and backtrace LUB estimation strategies. Each curve has an equal word-level beam and the individual data points on each curve correspond to varying state-level beams (equal to and less than the word-level beam).

approximation for each node is the maximum unigram probability of all words whose pronunciations pass through the node;

4. Exact LM incorporation — exact LM applied at nodes which contribute to a number of pronunciations below the LM incorporation threshold.

The approximations used in approaches (2) and (3) (see section 2.3) may be computed in advance and may be applied for those nodes above the LM incorporation threshold if approach (4) is used.

Figure 11 shows the effect of the context upper bound and unigram smearing approximations. These results indicate that unigram smearing is an effective LM approximation, whereas the context upper bound has no appreciable effect (and is even counter-productive at higher pruning levels). At the cost of an additional number of search errors, the size of the active search space (and hence the running time) is reduced by a factor of two or more when unigram smearing is applied. Since the unigram LM approximation at the state level can be an inaccurate estimate (eg when an infrequent word occurs in a specific context), the best tradeoff between search error and active model count occurs when the state-level beam is a little larger than the word-level beam (in contrast to the previous experiments where better performance was observed when the state-level beam was smaller than the word-level beam).

Figures 12 and 13 show the performance of incorporating exact LM probabilities at the state-level for those nodes that fall below the LM incorporation threshold. Two methods for dealing with nodes above the incorporation threshold – no LM and uniform smearing – were evaluated. Figure 12 indicates that there is a consistent 1–2% search error due to unigram smearing, as well a gradual increase of up to an extra 4% search error as the LM is applied to an increasing number of nodes.

Figure 13 shows how the computational cost varies with the LM incorporation threshold in the cases when the unigram LM approximation is or is not applied to those nodes above the threshold. Two measures of computational cost are used: the active model count and the running time. The lower curve of each graph plots the active model count; the upper curve the running time. As expected, the active model count decreases as more nodes in the pronunciation prefix tree use exact LM bounds. However, there is not a concomitant reduction in running time. In the case where where unigram smearing is not applied to those nodes above the threshold, LM incorporation is worthwhile at the state level, with the minimum in running time being at an LM incorporation threshold of about 80 pronunciations per node. Although this level of incorporation covers over 99.9% of all nodes in the tree, those which are incorporated in over 80 pronunciations are predominantly the first and second phones of a pronunciation and are hence activated more often. On average, around 22% of active nodes are above this LM incorporation threshold. The same tradeoff applies when unigram smearing is applied to those nodes that do not incorporate exact LM bounds. In this case the unigram approximation to the exact LM upper bound is sufficiently good, resulting in a very small improvement in active model count resulting from exact LM incorporation. In this case, running time reaches

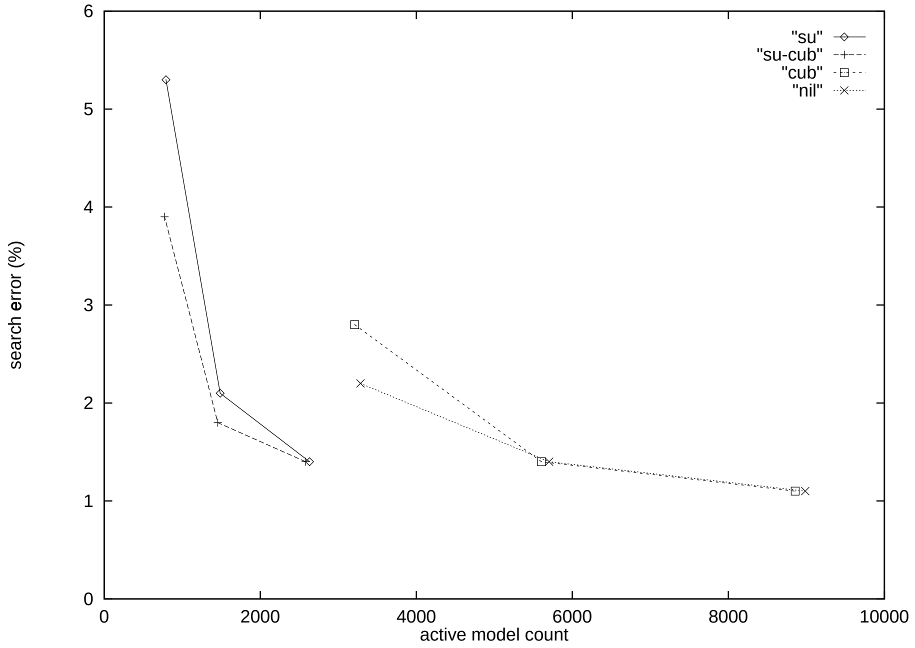


Figure 11: Graph of search error vs. active model count for different state-level language model approximations. “su” uses unigram smearing, “cub” uses the context upper bound method,“su-cub” uses both (added in log domain) and “nil” uses no state-level incorporation of LM information.

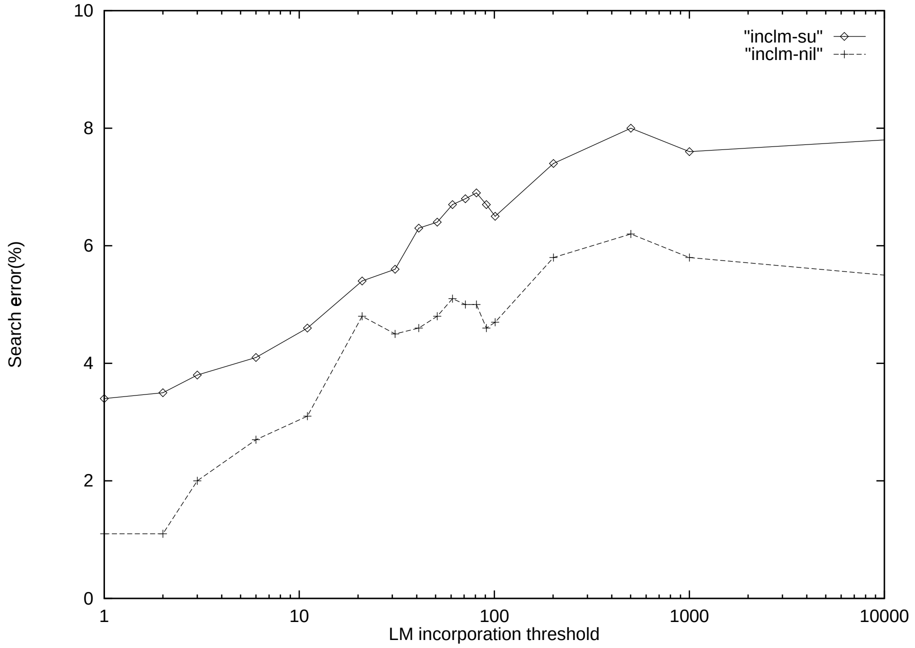
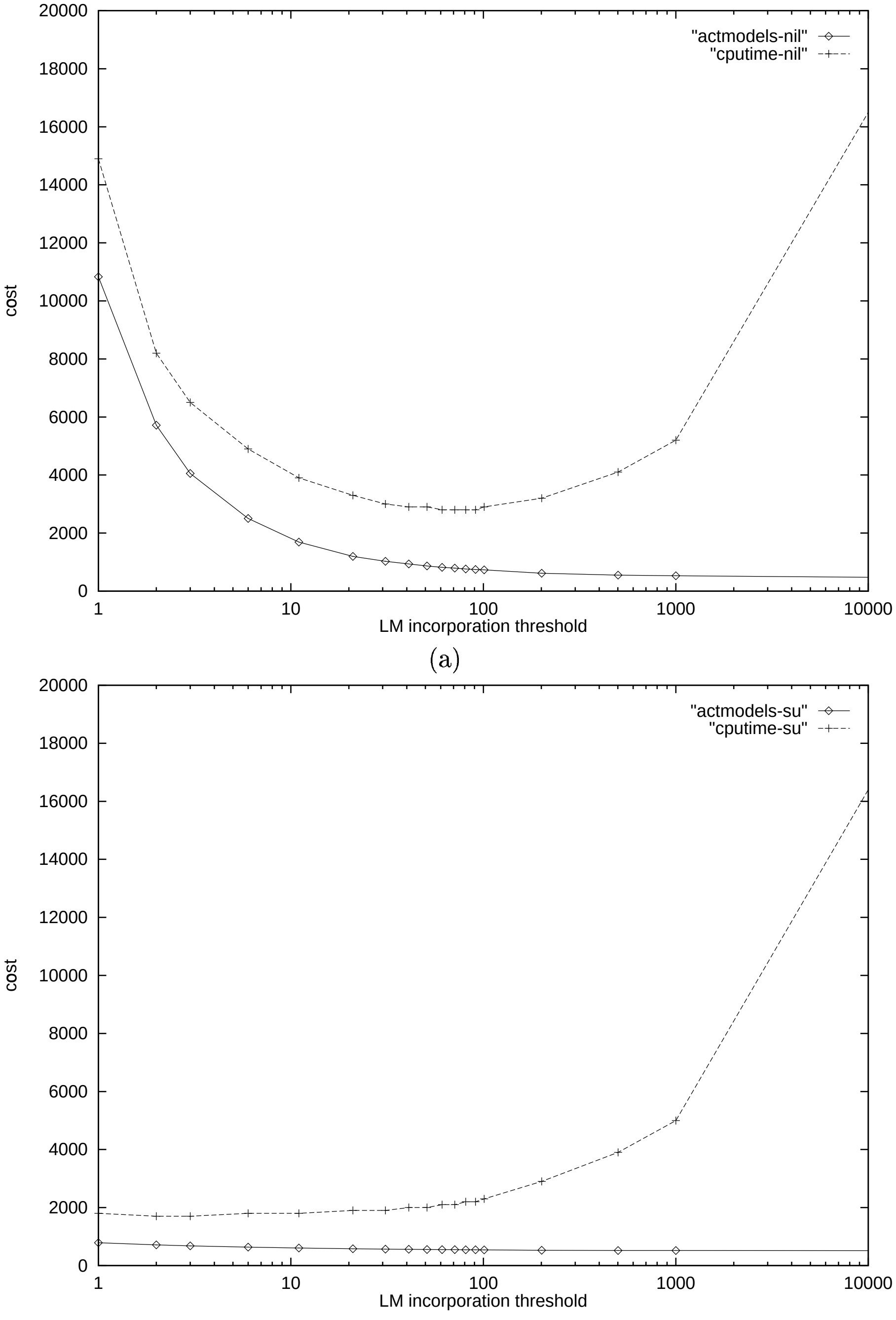


Figure 12: Graph of search error vs. LM incorporation threshold when the exact LM probabilities are applied at the state level. The labels “inclm-su” and “inclm-nil” indicate that the unigram smearing and no LM techniques, respectively, have been applied to the state level nodes above the threshold.



(a)

(b)

Figure 13: Graph of "computational cost" versus degree of LM incorporation. (a) No LM information is used at the state level for nodes which contribute to a number of pronunciations that is greater than the LM incorporation threshold; (b) the unigram approximation is used when the exact LM probabilities are not incorporated at the state level. On each graph, the lower curve corresponds to the number of active models used and the upper curve is the CPU time (1000 is realtime on an Ultra-167).

a minimum at an LM incorporation threshold of 1, when exact LM probabilities are used only for those nodes which contribute to the pronunciation of a single word.

4.4 Extensions

As mentioned in section 2.3, it is possible to perform a decoding with a larger acoustic model vocabulary than LM vocabulary, so not every word represented in the pronunciation tree need be in the LM provided there is some mechanism for specifying (default) probabilities for "unknown" words. One approach is to include a general "unknown" word class within the LM, although more sophisticated approaches may make use of word classes produced by some form of tagger (statistical, semantic or syntactic). We have performed an initial experiment using a system trained on the British English database WSJCAM0 [35] using a 20K language model (the 1994 DARPA NAB language model) and a 357,000 word pronunciation dictionary (the British English Example Pronunciation (BEEP) dictionary[8]). The 337,000 words not in the language model were mapped to the unknown symbol (*UNK*) and the LM probabilities of *UNK* (given a context) were scaled by the number of unigrams. Although the resultant pronunciation tree was 20 times larger compared with the 20K vocabulary, the average number of active nodes increased by a factor of 3.6. This "proof-of-concept" experiment resulted in a relative search error increase of 6%. This result is encouraging since it indicates the search algorithm can clearly scale favourably with vocabulary size. Note that there are clear improvements that could be made to the language model and pronunciation dictionary if we were developing a complete 350,000 word system.

This algorithm is well suited to the generation of word graphs. Unlike some Viterbi-based approaches (e.g., [36, 5]), this algorithm does not use an estimate of the most probable boundary between two words. While this results in a lower probability of search errors (although the evidence suggests that the word pair boundary approximation is not a significant contributor to search errors), it also results in word graphs of a higher density. Typical word graphs produced by this algorithm have a density of 50-100 compared to 10-20 in those produced by the approach that uses optimal word-pair boundaries [5]. However, word graphs produced by this algorithm may be further pruned. Using this start-synchronous stack-based search architecture, less constrained approximations than the word-pair may be applied with little extra computational cost.

5 CONCLUSION

We have presented a search architecture and algorithm for large vocabulary continuous speech recognition. The start-synchronous stack-based search operates in a single pass and – because there is a simple, well-defined interface between

[8] Version 0.7, available by anonymous ftp from ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/dictionaries/beep.tar.gz.

the search and the language model – allows the application of arbitrary, long-span language models. Furthermore, extra information may be stored about a hypothesis (eg interword pauses) without affecting the language model state.

We have focused primarily on context-independent phone modelling, largely because the ABBOT system has very good performance using context-independent models. However, the algorithm can be immediately applied to context-dependent models. This has been done for within-word context-dependent models [26] and the method used by Ravishankar [30] for cross-word context-dependent models may be applied in this case.

We have also developed pruning techniques that take advantage of specific features of the connectionist/HMM approach. Phone deactivation pruning uses the local posterior probabilities estimated directly by the recurrent network, pruning those phones with a local posterior probability below a threshold. This approach can reliably prune around 64% of the phones, resulting in a speed improvement of a factor of seven (or more) with a search error of about 1%. Recent work by Willett et al. [37] has extended the phone deactivation approach to "traditional", likelihood-based HMM systems, in which the local state posterior probabilities are estimated using an efficiently computed estimate of $p(\mathbf{x})$.

In summary we have developed an efficient search algorithm, specifically tuned to the connectionist/HMM approach, that enables close to realtime decoding of large vocabulary continuous speech recognition problems with minimal search error.

ACKNOWLEDGMENTS

We are grateful to Tony Robinson, with whom we have had many insightful conversations about the work reported here. Language models and pronunciation dictionaries were provided by Carnegie Mellon University and the International Computer Science Institute. This work was supported by ESPRIT Long Term Research Project 20077 (SPRACH) and ESPRIT Basic Research Project 6487 (WERNICKE).

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 179–190, 1983.

[2] A. B. Poritz, "Hidden Markov models: a guided tour," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, (New York), pp. 7–13, 1988.

[3] G. D. Forney, Jr., "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, 1973.

[4] T. K. Vintsyuk, "Element-wise recognition of continuous speech composed of words from a specified dictionary," *Kibernetika*, vol. 7, pp. 133–143, 1971.

[5] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, pp. 43–72, 1997.

[6] B. Lowerre and R. Reddy, "The Harpy speech understanding system," in *Trends in Speech Recognition* (W. A. Lea, ed.), Prentice Hall, 1980.

[7] H. Ney, D. Mergel, A. Noll, and A. Paesler, "Data-driven search organisation for continuous speech recognition," *IEEE Trans. Signal Processing*, vol. 40, pp. 272–281, 1992.

[8] J. J. Odell, *The Use of Context in Large Vocabulary Speech Recognition.* PhD thesis, University of Cambridge, 1995.

[9] R. Schwartz, L. Nguyen, and J. Makhoul, "Multiple-pass search strategies," in *Automatic Speech and Speaker Recognition – Advanced Topics* (C.-H. Lee, F. K. Soong, and K. K. Paliwal, eds.), ch. 18, pp. 429–456, Kluwer Academic, 1996.

[10] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM Journal of Research and Development*, vol. 13, pp. 675–685, 1969.

[11] N. J. Nilsson, *Problem Solving Methods of Artificial Intelligence.* New York: McGraw-Hill, 1971.

[12] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Information Theory*, vol. 21, pp. 250–256, 1975.

[13] L. R. Bahl and F. Jelinek, "Apparatus and method for determining a likely word sequence from labels generated by an acoustic processor." US Patent 4,748,670, May 1988.

[14] P. S. Gopalakrishnan, L. R. Bahl, and R. L. Mercer, "A tree-search strategy for large vocabulary continuous speech recognition," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 1, (Detroit), pp. 572–575, 1995.

[15] D. B. Paul, "An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 1, (San Francisco), pp. 25–28, 1992.

[16] P. Kenny, R. Hollan, V. N. Gupta, M. Lennig, P. Mermelstein, and D. O'Shaughnessy, "A*-admissible heuristics for rapid lexical access," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 59–58, 1993.

[17] F. K. Soong and E.-F. Huang, "A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 1, (Toronto), pp. 705–708, 1991.

[18] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions and substitutions with applications to speech recognition," *IEEE Trans. Information Theory*, vol. 21, pp. 404–411, 1975.

[19] M. M. Hochberg, S. J. Renals, A. J. Robinson, and G. D. Cook, "Recent improvements to the ABBOT large vocabulary CSR system," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, (Detroit), pp. 69–72, 1995.

[20] T. Robinson, M. Hochberg, and S. Renals, "The use of recurrent networks in continuous speech recognition," in *Automatic Speech and Speaker Recognition – Advanced Topics* (C. H. Lee, K. K. Paliwal, and F. K. Soong, eds.), ch. 10, pp. 233–258, Kluwer Academic Publishers, 1996.

[21] H. Bourlard and N. Morgan, *Connectionist Speech Recognition—A Hybrid Approach.* Kluwer Academic, 1994.

[22] A. J. Robinson, "The application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, 1994.

[23] J. Hennebert, C. Ris, H. Bourlard, S. Renals, and N. Morgan, "Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems," in *Proc. Europ. Conf. Speech Communication and Technology*, (Rhodes, Greece), pp. 1951–1954, 1997.

[24] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge MA: MIT Press, 1990.

[25] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 2, pp. 161–175, 1994.

[26] D. J. Kershaw, M. M. Hochberg, and A. J. Robinson, "Context-dependent classes in a hybrid recurrent network-HMM speech recognition system," in *Advances in Neural Information Processing Systems*, vol. 8, MIT Press, 1996.

[27] V. Gupta, M. Lennig, and P. Mermelstein, "Fast search strategy in a large vocabulary speech recognizer," *J. Acoust. Soc. Amer.*, vol. 84, pp. 2007–2017, 1988.

[28] L. R. Bahl, S. V. De Gennaro, P. S. Gopalakrishnan, and R. L. Mercer, "A fast approximate acoustic match for large vocabulary speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 1, pp. 59–67, 1993.

[29] R. Haeb-Umbach and H. Ney, "Improvements in beam search for 10 000-word continuous-speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 2, pp. 353–356, 1994.

[30] M. K. Ravishankar, *Efficient Algorithms for Speech Recognition.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1996.

[31] H. Bourlard, B. D'hoore, and J.-M. Boite, "Optimizing recognition and rejection performance in wordspotting systems," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 1, (Adelaide), pp. 373–376, 1994.

[32] X. Aubert, C. Dugast, H. Ney, and V. Steinbiss, "Large vocabulary continuous speech recognition of Wall Street Journal data," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 2, (Adelaide), pp. 129–132, 1994.

[33] S. Renals, "Phone deactivation pruning in large vocabulary continuous speech recognition," *IEEE Signal Processing Lett.*, vol. 3, pp. 4–6, 1996.

[34] P. S. Gopalakrishnan, D. Nahamoo, M. Padmanabhan, and M. A. Picheny, "A channel-bank-based phone detection strategy," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, vol. 2, (Adelaide), pp. 161–164, 1994.

[35] T. Robinson, J. Fransen, D. Pye, J. Foote, and S. Renals, "WSJCAM0: A British English speech corpus for large vocabulary continuous speech recognition," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, (Detroit), pp. 81–84, 1995.

[36] R. Schwartz and S. Austin, "A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, (Toronto), pp. 701–704, 1991.

[37] D. Willett, C. Neukirchen, and G. Rigoll, "Efficient search with posterior probability estimates in HMM-based speech recognition," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, (Seattle), pp. 821–824, 1998.