CONTEXT-DEPENDENT SEARCH IN A CONTEXT-INDEPENDENT NETWORK

Fabio Brugnara

ITC-irst – Centro per la Ricerca Scientifica e Tecnologica
I-38050 Povo di Trento, Italy

ABSTRACT

This paper introduces a search algorithm for continuous speech recognition, working on a network that integrates both lexical and linguistic constraints. It differs from traditional Viterbi beam-search in that it does not assume that the network includes any information regarding context dependency of the acoustic models. Phonetic context dependency is instead taken into account by the search procedure itself, in a way that uniformly deals with within-word and cross-word contexts. In the paper the algorithm is described in detail, and results are given on two representative tasks: American English dictation and Italian broadcast news.

1. INTRODUCTION

The use of context-dependent acoustic models is of primary importance to achieve high accuracy in speech recognition. The integration of context dependency in the decoding process is straightforward when limited to the consideration of within-word contexts, as in this case it only amounts to appropriately transcribing the words as sequences of context-dependent units. On the other hand, if contexts have to be considered at word boundaries, special care is needed at word transitions, because in this case the transcription of a word depends on surrounding words. In the common case in which the main structure searched by the decoder is a network of models, either completely compiled beforehand or dynamically expanded on demand, this requires a multiplication of arcs at word boundaries in order to properly join contexts. This problem can be dealt with through composition of Weighted Finite State Tranducer (WFST), as proposed e.g. in [1], but the intermediate steps are very resource demanding, and the network structure depends on the particular set of models used.

The algorithm presented here is instead aimed at decoupling context dependency from the network structure, that maintains the context-independent description of words, and uniformly deal with within-word and cross-word contexts. Given the need for conciseness, the description must assume some familiarity of the reader with the terminology in beam-search decoding. An excellent survey of decoding techniques can be found in [2].

2. OVERVIEW OF THE STANDARD VITERBI DECODER

To ease the understanding of the proposed algorithm, this section gives an outline of the standard Viterbi beam-search in the ITC-irst recognizer. Following the definitions in [1], the recognition network can be seen as a WFST, where input labels correspond to phonetic units, and output labels, attached to word-end arcs, correspond to words. The decoder assumes that input labels are in one-to-one correspondence with HMMs. In the following, arcs that carry an input label will be referred to as named arcs, those without any label will be called empty arcs.

This work was partially funded by the European Commission under the CORETEX project (IST-1999-11876)

The search algorithm keeps the whole network in memory, but dynamically allocates memory for the structures that depend on the decoder status, such as information concerning active nodes and active arcs. For each active named arc, a trellis column is kept, which is updated at each time frame with status information for each internal node of the corresponding HMM. For each active node, a status record is kept, containing the best likelihood for it so far, and backtrack information. All likelihood computation are done in the logarithmic domain. The operations at each time frame can be summarized as follows:

- The list of active nodes is pruned, removing nodes which have a likelihood below the current beam-search threshold.
- The list of active nodes is scanned to activate the outgoing named arcs that are not already active.
- The list of active arcs is scanned to propagate the paths within the HMMs. The maximum likelihood value is kept, to be used for setting the beam threshold for the next time frame. The likelihood of each HMM final state is assigned as output score to the corresponding arc.
- The output scores of arcs are combined to update status information for the network nodes, and a new beam threshold is set.
- The list of active nodes is scanned for propagating scores along empty arcs.
- The backtrack information for all the “relevant” nodes for the current time frame are stored while doing recombination. The set of relevant nodes can include all nodes, or, more commonly, only the nodes reached by a word-end arc.

3. THE CONTEXT-DEPENDENT SEARCH

The procedure for the context-dependent search is similar to the one described above. In this case however, status information and score propagation are done so as to keep separate status information for different contexts. The main structure is outlined here, and described in more detail in the following subsections. Table 1 explains the symbols used.

Instead of being associated to a single HMM, a named arc $\tau : s(\tau) \xrightarrow{u(\tau)} d(\tau)$ is associated to a subnetwork $\mathcal{N}(\tau)$ with several entry nodes and exit nodes. An entry node exists for each seen left context of the source node $s(\tau)$, and an exit node exists for each right context of the destination node $d(\tau)$. Arcs in $\mathcal{N}(\tau)$ are associated to the HMMs corresponding to the unit $u(\tau)$ in different contexts, and are connected to nodes in $\mathcal{N}(\tau)$ according to the context they represent (see Figure 1).

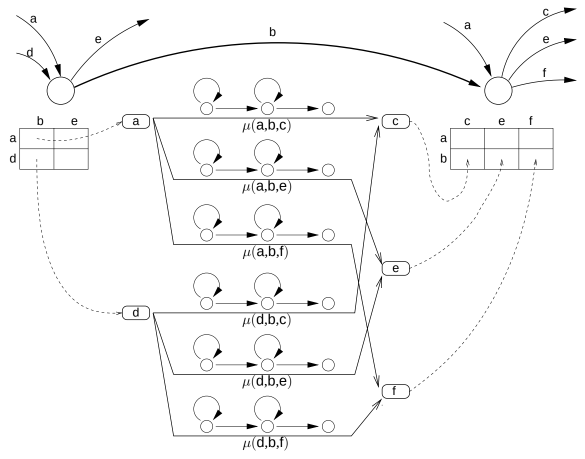

Fig. 1. *The subnetwork $\mathcal{N}(\tau)$ associated to an arc τ with label* b.

N,E	The sets of named and empty arcs, respectively
$\mathcal{N}(\tau)$	The subnetwork of HMMs associated to arc τ
L_s	The set of left context of hypotheses that actually entered node s
R_s	The complete set of right contexts for a node s
L_τ	The set of left context of hypotheses that actually entered arc τ
$p_s(l,r)$	The status information corresponding to left context l and right context r in node s
$\Pr(\tau)$	The log-probability of arc τ
$p_\tau(r)$	The output likelihood of arc τ for right context r
$\mu(l,b,r)$	The HMM to be used for modeling unit b in left context l and right context r

Table 1. *Symbols used in the description of the algorithm.*

A node s keeps a matrix p_s of status information, with an entry for each possible context combination $(l,r) \in L_s \times R_s$. The set R_s for each node is computed beforehand, by examining the labels on outgoing arcs, while the set L_s is incrementally updated when new hypotheses enter the node.

When expanding an arc, input scores are taken from the proper column of the score matrix of the source node and propagated into the internal trellises of the models according to their left context.

The likelihoods for the entry nodes of $\mathcal{N}(\tau)$ are set to:

$$I_\tau(l) = p_{s(\tau)}(l, u(\tau)) + \Pr(\tau), \forall l \in L_\tau$$

All the HMM trellises are then updated processing the current observation, and their outputs assigned according to their right contexts. After all arc hypotheses have been scored, the likelihoods for each node s are updated so as to have, $\forall (l,r) \in L_s \times R_s$:

$$p_s(l,r) = \max\left\{p_\tau(r) \colon \tau \in N, d(\tau) = s, u(\tau) = l\right\}$$

and backpointers are updated accordingly. Score propagation is then performed along empty arcs, so as to have, for each node s and $(l,r) \in L_s \times R_s$:

$$p_s(l,r) = \max\left\{p_{s(\tau)}(l,r) + \Pr(\tau) \colon \tau \in E, d(\tau) = s\right\}$$

3.1. Node pruning

For each active node s and $(l,r) \in L_s \times R_s$, $p_s(l,r)$ is compared with the current threshold, and set to $-\infty$ if it is lower. A node is removed if all its associated likelihoods are deleted in this way.

3.2. Arc activation

For each active node s, the outgoing arcs are examined. An arc τ is activated if any of the likelihoods $\{p_s(l, u(\tau)) + \Pr(\tau), l \in L_s\}$ is greater than the current threshold.

3.3. Named arc expansion

The paths of the network nodes are then propagated into the HMMs. For each active arc, $\tau : s \xrightarrow{u} d$, the following steps are performed:

- If the set L_s contains any element $l' \notin L_\tau$, L_τ and $\mathcal{N}(\tau)$ are expanded. For this purpose, a new entry node for l' is added to $\mathcal{N}(\tau)$, the context-mapping module is queried to get the model corresponding to each triphone $\{(l', u, r) \colon r \in R_d\}$, and arcs are added to $\mathcal{N}(\tau)$ with the appropriate models, joining the new entry node with the exit nodes.
- For each $l \in L_\tau$ such that $p_s(l,u) > -\infty$, the corresponding entry node in $\mathcal{N}(\tau)$ is initialized with $p_s(l,u) + \Pr(\tau)$.
- Paths are then propagated within $\mathcal{N}(\tau)$ with a step of the Viterbi algorithm, using the HMM parameters and the current frame of acoustic parameters. An arc is deactivated if all node likelihoods in $\mathcal{N}(\tau)$ are below the current threshold before expansion.
- The output scores of the HMMs are used to update the exit nodes of $\mathcal{N}(\tau)$. These likelihoods $\{p_\tau(r) \colon r \in R_d\}$ are the output of the arc for the current frame, and will be used when doing path recombination on the main network.

In propagating scores within $\mathcal{N}(\tau)$, redundant computation is avoided if more instances of the same model are attached to different arcs leaving the same node.

When doing path expansion, the maximum value of all computed node likelihoods is computed, to be used later for setting the beam threshold.

3.4. Arc score recombination

When named arcs have been processed, their scores are used to update the status of network nodes, after clearing the list of active nodes. Based on the maximum likelihood values computed in the previous step, two thresholds ω_w and ω_i are computed, to be used in pruning word-end nodes and word-internal nodes, respectively. For each arc $\tau : s \xrightarrow{u} d$, and for each $r \in R_d$:

- If $p_\tau(r) \le \omega$, context r is skipped, otherwise the following actions are taken. Here ω is the proper threshold between ω_w and ω_i.
- Node d is activated if not already active.
- If $u \notin L_d$, u is added to L_d.
- If $p_\tau(r) > p_d(u,r)$, then $p_d(u,r)$ is replaced by $p_\tau(r)$, and the backpointer is updated accordingly.

3.5. Empty arc expansion

Empty arc expansion is based on a queue, initialized with all active nodes which have some outgoing empty arc. Then, the following procedure is repeated until the queue empties:

- The first element s is taken from the queue.
- The score is propagated through all empty arc leaving s. If the likelihood of a node d is updated, and d has outgoing empty arcs, d is put on the queue.

Score propagation for empty arc is trivial in the classical case, but it is more complex in the context-dependent search. For each empty arc $\tau : s \longrightarrow d$, for each $l \in L_s$, and for each $r \in R_d$:

- If $p_s(l,r) + \Pr(\tau) \le \omega$, the combination (l,r) is skipped.
- Node d is activated if not already active.
- If $l \notin L_d$, l is added to L_d.
- If $p_s(l,r) + \Pr(\tau) > p_d(l,r)$, then $p_d(l,r)$ is replaced by $p_s(l,r) + \Pr(\tau)$, and the backpointer is updated accordingly.

Notice that if there exists an empty arc $\tau : s \longrightarrow d$, then it must be $R_d \subseteq R_s$.

3.6. Storage of backpointers and backtracking

When status information about all nodes has been updated, backpointers are stored for later use in recovering the best path.

At each time instant t, a backpointer is kept for each node s and each pair $(l,r) \in L_s \times R_s$ for which $p_s(l,r) > -\infty$. The backpointer itself links to a network node and a previous time index, and also includes indication of the particular context combination on that node through which the best path reaching s passed.

After all time frames have been processed, backtracking is performed as usual by following the backpointers stored during the search, starting from the node with the highest likelihood. The only difference comes from the additional context information carried by backpointers, as explained above.

3.7. Triphone-to-model mapping

As previously stated, the mapping $(l,b,r) \mapsto \mu(l,b,r)$ from triphones to models is done at run time. If the last unit l of a path entering an arc τ has not been already seen on τ , a module is queried to know which model has to be used for each triphone $(l,u(\tau),r)$, $r \in R_{d(\tau)}$. This module is external to the decoder and can therefore implement different policies. This information is cached by the decoder, so that a mapping for a given triphone is asked only once. In the experiments, for example, where the models are built using a Phonetic Decision Tree (PDT), the mapping is obtained by querying the decision tree itself.

The triphone-mapping module can notify the decoder when certain units are to be considered inherently context-independent, as is often the case for the silence unit or other filler units. When expanding and combining arcs labeled by such units, the decoder takes advantage of it by not building subnetworks and disregarding context dependency when propagating likelihoods.

4. EXTENSIONS AND IMPROVEMENTS

4.1. Arc chains

It is common, when compiling LM networks, that sequences of arcs form a *chain*, i.e. they link nodes having a single entry arc and a single exit arc. A chain can be conveniently represented by a single arc labeled by a sequence of units, thereby reducing the network size. Of course, such arcs are to be processed appropriately when doing arc expansion.

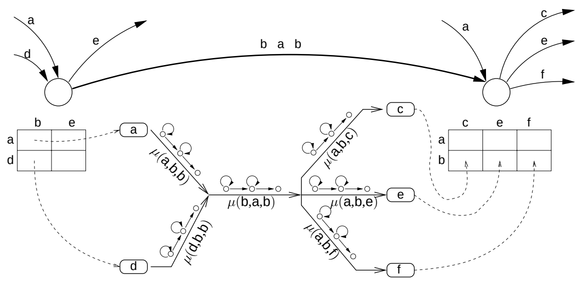

Fig. 2. *The subnetwork on an arc representing a chain* b a b.

In the case of the context-dependent search, however, it is already the common case that an arc is associated to a subnetwork $\mathcal{N}(\tau)$, therefore the processing for the inner path expansion is the same as for other arcs. The only differences are the topology of $\mathcal{N}(\tau)$, and the rules used when expanding it on demand. Figure 2 shows the typical structure for these arcs.

When activating an arc τ for the first time, the portion of $\mathcal{N}(\tau)$ that do not includes the entry nodes can be built at once. Then, for each left context l entering the arc, the triphone-mapping module has to be queried to get $\mu(l,u_1,u_2)$ only, where u_1,u_2 are the first and the second unit in the chain, respectively. That is, only a single arc is added to $\mathcal{N}(\tau)$, unlike to what happens with normal arcs.

Moreover, in this case it is simple to detect cases in which entry arcs in $\mathcal{N}(\tau)$ are redundant, since all entry arcs converge.

4.2. Improving memory usage through memory mapping

To lessen the impact of large networks, some techniques have been integrated to reduce memory usage, taking advantage of virtual memory features commonly available on modern operating systems.

Memory-mapping of the network: By using a technique called memory-mapping, the operating system can be instructed to use a file as support for a particular region of the virtual address space of a process, instead of the swap space. The actual fetching of data from disk is under control of the operating system, with a granularity that depends on the system page size. If the region is read-only, as is the case of the network structure for the present algorithm, this allows to reserve a portion of virtual address space, without actually loading the pages unless they are referenced. Therefore, the usage of real memory is not determined by the whole size of the network, but by the network portion actually visited during search.

On-demand memory-mapping of network regions: If the memory mapping is done for the whole network at once, the virtual address space is still dependent on the total size. In order to reduce both the virtual address space and the real memory space, a demand-mapping of network regions has been introduced. With this option, the mapping of a network region is explicitly requested by the program only upon a reference to that region. The number of mapping requests can be reduced by reordering the network nodes. This can be easily done off-line by visiting the network in a breadth-first manner, that is with the same policy the decoder uses, and renumbering the nodes so as to reflect the order in which they are encountered. Since the information about nodes and arcs is sequentially stored in the network structure, this helps in improving

	Max. Mem.	*Avg. Mem.*	*RTR*
Baseline	351Mb	306Mb	3.9
Paged	270Mb	162Mb	4.0
Paged and Sorted	274Mb	147Mb	4.0

Table 2. *Decoder resource usage on the 20K WSJ '93 Evaluation task with different options enabled.*

the locality of references to network elements.

5. EXPERIMENTS

5.1. Wall Street Journal

The first experiment is based on the Wall Street Journal corpus, a widely used speaker independent American English dictation task. Acoustic models were trained on the standard SI-284 training set. The test set is the November '93 Evaluation Test, a set of 213 sentences uttered by 10 speakers. The language model is a 20k trigram backoff LM, defined in the evaluation specification. In the results of the evaluation, performances ranged from 11.7% to 19.0% Word Error Rate (WER), with an average of 14.9%.

The set of acoustic models included 27320 models, with a PDT-based tied-state architecture of 8873 tied states and 71115 Gaussians. The model complexity and training procedure for HMMs are similar to the WSJ system described in [3], that reports a 12.7% WER, except that our system used gender-independent models, and the lexicon made available by LIMSI after the '93 evaluation. This was done on purpose, because a baseline with predictable results was required to validate the approach. The compilation of the standard trigram LM resulted in a network with 5,428,622 nodes, 5,197,189 named arcs, and 9,660,367 empty arcs. The network contained 1,361,617 arc chains with an average length of 3 arcs. The resulting system produced a WER of 12.9%.

Table 2 shows the resource usage, and the effect of the techniques described in section 4.2, as measured on a 1.5GHz AMD Athlon CPU running Linux 2.4.9. Memory usage was measured by exploiting the `/proc/` interface exposed by Linux. Comparing the first and second row of the table, it can be seen that explicit demand paging of network regions is effective in mitigating the drawbacks of the static representation. In the last row, the effect is shown of reordering the nodes in order to reduce mapping requests, which adds another small improvement. Running times are not significantly affected by introducing paging, as expressed by the Real Time Ratio (RTR) in the last column.

5.2. Italian broadcast news

At ITC-irst there is an ongoing activity on Italian Broadcast News transcription, that has been described in previous papers [4]. The transcription system includes several stages, but here we focus on a single step of recognition with speaker independent models, and results are presented to compare the performances of the within-word and cross-word systems. Training data consist of about 57 hours of speech coming from radio and TV newscast, including planned and spontaneous speech, which constitute the wideband portion of the ITC-irst BN corpus. The test data include the wideband portion of 6 radio shows and 2 TV shows, for a total of 1 hour and 24 minutes of speech.

The within-word models of the baseline system are built with an agglomeration technique [5] using phonetically tied mixtures, and correspond to those used in our best system so far. Previous attempts to use PDT-based state tying on this task with within-word models did not result in any improvement. Given the good performance of PDT tying when combined with cross-word models on WSJ data, however, it was decided to use the same procedure used for the WSJ experiments in building the cross-word acoustic models. Table 3 summarizes the characteristics of the two model sets. Given the different tying scheme, they are not directly comparable, but the overall complexity is similar.

Model set	*Models*	*Mixtures*	*Gaussians*
Within-word	9,734	36,671	42,212
Cross-word	8,458	7,509	60,007

Table 3. *Characteristics of the model sets used in BN experiments.*

	Max. Mem.	*Avg. Mem.*	*RTR*	*WER*
Baseline	642Mb	538Mb	7.1	15.2
Paged and Sorted	500Mb	263Mb	7.5	”
Within-word	575Mb	425Mb	6.4	16.0

Table 4. *Decoder performance and resource usage on the 64K BN task. The last row refers to the reference within-word system.*

The language model is an interpolated trigram LM with a 64k word vocabulary, trained on a corpus mostly composed by newspaper texts, augmented with the transcriptions of the BN data used in training acoustic models. The network obtained after LM compilation contained 9,047,564 nodes, 8,713,951 named arcs, and 19,974,485 empty arcs. There are 1,550,862 arc chains in the network, with an average length of 2.9 arcs.

Table 4 shows the performance and resource usage of the cross-word and within-word systems. In comparing memory usage, one must take into account that the within-word system has not yet been updated with on-demand mapping of network regions.

6. CONCLUSION AND FUTURE WORK

In this paper, a search algorithm for speech recognition has been described, using a network that integrates both linguistic and lexical information, but is independent from the context dependency characteristics of the acoustic units. To demonstrate the feasibility of the approach, results have been presented on two representative tasks. Future work will be devoted to improve the algorithm with respect to run time and memory usage.

7. REFERENCES

[1] M. Mohri, M. Riley, D. Hindle, A. Ljolje, and F. Pereira, "Full expansion of context-dependent networks in large vocabulary speech recognition," in *Proc. of ICASSP*, Seattle,WA,USA, 1998, pp. 665–668.

[2] X. L. Aubert, "An overview of decoding techniques for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 89–114, 2002.

[3] J. J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph.D. Thesis, Cambridge University, 1995.

[4] F. Brugnara, M. Cettolo, M. Federico, and D. Giuliani, "Advances in automatic transcription of broadcast news," in *Proc. of ICSLP*, Beijing, China, 2000, pp. II:660–663.

[5] F. Brugnara, "Model agglomeration for context-dependent acoustic modeling," in *Proc. of EUROSPEECH*, Aalborg, Denmark, 2001, pp. 1641–1644.