A SCALABLE SYSTEM FOR EMBEDDED LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

G. Linarès, D. Massonié, P. Nocéra, C.Lévy

Laboratoire Informatique d'Avignon (LIA)
University of Avignon, France
georges.linares,dominique.massonie,christophe.levy,pascal.nocera@univ-avignon.fr

ABSTRACT

This paper presents a system for large vocabulary continuous speech recognition in condition of constrained hardware resources. We investigate efficient pruning and caching strategy aiming to handle extensive acoustic and linguistic modeling. Software components are analyzed in terms of resource consuming. Then, we evaluate the system performance in extreme configuration where acoustic and linguistic models are dramatically pruned. Results show that the system design we proposed allows to use large HMM-based acoustic models and trigram language models while performing very fast decoding, about 0.6 real-time on a standard desktop computer while remaining the transcript relevance.

1. INTRODUCTION

Embedded Large Vocabulary Continuous Speech Recognition (LVCSR) is one of the most promising application of speech processing. The last decade, the speech technology was largely integrated in mobile systems as hardware performance was strongly improved. Nevertheless, light recognition systems offer usually restricted speech-based services, such as name dialing or voice command. Moreover, the embedding of LVCSR engine implies a specific design in which the models precision are degraded and the search algorithm is strictly pruned, in comparison to the ones involved in unconstrained systems. The fast improvement of hardware capacities may lead soon to an integration of more powerful Automatic Speech Recognition (ASR) system into light and mobile devices. Nevertheless, recent proposals of embedded systems require still some software adaptation and rigorous pruning of models involved in recognition process.

In this paper, we investigate methods for reducing the resource required for LVCSR while preserving the functional model of the transcription machine. We focus on acoustic and linguistic resource management which allows to handle both large acoustic and linguistic models.

The next section of this paper present the LIA LVCSR system. We describe firstly the search algorithm. Then, we focus on acoustic and linguistic handling. A fast acoustic manager is described and we propose an efficient caching technique for fast access to trigram probabilities.

The third section presents the evaluation we achieved on large vocabulary tasks. Experiments are carried out in the framework of the ESTER [3] evaluation campaign.

Lastly, we conclude and suggest some key-points to better improve the tradeoff between accuracy and resource consumption in LVCSR systems.

2. SPEERAL DECODER

2.1. Search Strategy

The search engine of Speeral toolkit is an A*-based decoder. A* is an algorithm dedicated to the search of the best path in a graph. It has been used in several speech recognition engines, generally for word-graph decoding. In Speeral, the search algorithm operates on a phoneme lattice, which are estimated by using cross-word and context-dependent HMM.

The exploration of the graph is supervised by an estimate function $F(h_n)$ which evaluates the probability of the hypothesis crossing the node n:

$$F(h_n) = g(h_n) + p((h_n)$$

where $g(h_n)$ is the probability of the current hypothesis which results from the partial exploration of the search graph (from the starting point to the current node n); $p(h_n)$ is the probe which estimates the probability of the best hypothesis from the current node n to the ending node. The stack of hypotheses is ordered on each node according to $F()$. The best paths are then explored firstly. This depth first search refines the evaluation of the current best hypothesis and low-probability paths are cut-off,

leading to search backtrack. It is clear that precision of the probe function is a key point for search efficiency. We have produced substantial efforts to improve the probe by integrating, as soon as possible, all available information.

The Speeral look-ahead strategy is described in the next section.

2.1. Acoustic-linguistic look-ahead

As explained previously, the probe function aims to evaluate the probability of each path which may be developed. The more this approximation is close to the exact one, the sooner a decision of leaving or developing a path is taken. Moreover, the CPU-cost of this function is critical as it is used at each node of the search graph. We use a long-term acoustic probe combined with a short-term linguistic look-ahead. The acoustic term is computed by a Viterbi-back algorithm based on context-free acoustic models. This algorithm evaluates the best acoustic probabilities from the end-point to the current one. Of course, the evaluation of all partial paths are performed once, in a first pass. Nevertheless, for search consistency, the probe must always provide an upper limit of path probabilities. So, the best phoneme sequences are rescored by using upper-models. Upper models are context-free models resulting from the aggregation in a large HMM, of all the context-dependent states which were associated to a context-free model, remaining left-right transition constraints. Hence, the probability of emission given the upper-model is an upper-limit of path probabilities.

Anticipating the linguistic information (known as LMLA - Language Model Look-Ahead) enables the comparison of competing hypotheses before reaching a word boundary. The probability of a partial word corresponds to the best probability in the list of words sharing the same prefix:

$$P(W^*|h) = \max_i P(Wi|h)$$

where W* is the best possible continuation word and h the word history (partially present in g(hn)). The lexicon is stored as a PPT (Pronunciation Prefix Tree), see figure 1. Each tree node contains the list of reachable words. To ensure the consistency between linguistically well formed hypotheses and pending ones, linguistic probabilities have to be computed at the same n-gram order. This means doing LMLA also at the 3-gram level. We developed a fast computation and approximation method based on a divide and conquer strategy ([6]). Our approach consists in first comparing the list W*with the list of available trained 3-gram stored in the LM. The LM is an on-disk tree structure containing lists of word probabilities at each n-gram level. Comparing lists at runtime spares most of the LM back-off computation with low overhead (lists size are shown on table 1). The LMLA approximation does not affect the results. Moreover we introduced pre-computed LMLA probabilities to speed-up the computation of the biggest lists.

2.3. Lexicon structure and compression

The lexicon is stored as a Pronunciation Prefix Tree (PPT) for search efficiency [1]. Each node s (as a state reached during the decoding process) knows the wordlist of reachable words W (s) (see figure 1). The wordlist size decreases along the PPT to reach one single word at leaves. No other information like sub-tree dominance [7] is used. Due to pronunciation variation or tree scarcity, different nodes can share the same wordlist. As shown on table 2 most nodes are leaves. On average each wordlist tend to be shared by 3 internal nodes. Other properties of the lexical tree compression are also discussed in [8].

When the node is a leaf or due to PPT compression (figure 1) W (s) can contain one single word. The LMLA computation is then the same as a simple n-gram access, thus LMLA prefetches n-gram probabilities into the n-gram cache.

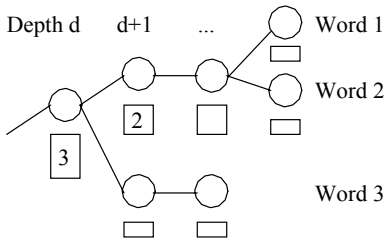

Fig. 1. *A sample end of Pronunciation Prefix Tree. Each node s contains the wordlist W (s) of reachable words. PPT compression is possible, for example the wordlists of word3 last two nodes are identical. Moreover if word1 and word2 are two different pronunciation of the same word, then more wordlists could be reduced and shared earlier in the tree, from depth d + 1.*

Table 1: *count of wordlist according to size. Most wordlists are smaller or equal to 5 words*

size	all	>5	>45	>98	>1000	>1500
27k	191k	7k	402	-	15	9
65k	403k	16k	-	402	34	23

Table 2 *: count of nodes and wordlists. The compression ratio of wordlists to (total # of nodes - leaves) / # of unique wordlist, which is about 3.*

nodes & wordlist	lexicon 27k	lexicon 65k
Total # nodes	191k	403k
Nodes without list (leave)	140k	278k
# of unique lists	**18k**	**43k**
# of duplicate lists	10396	26415
More than duplicate lists	5752	14837

2.4. Acoustic Handler

5.1.1. Motivation

The a priori estimation of the contribution of each component in the decoding duration is difficult, as it depends on the search strategy and on the models complexity. Nevertheless, considering the com- plexity of acoustic models involved in LVCSR systems, the computation of acoustic probabilities may take a large part of the decoding computational cost. [4] estimates this ratio ranges between 30% and 70% in a large vocabulary system. More recent systems use models of millions of parameters; such complexity should not be tractable without any fast calculation method. We produced substantial ef- forts in optimizing the acoustic management, by using an efficient caching scheme and an original method for fast-likelihood computation. A* decoding and state tying lead to an asynchronous use of acoustic probabilities, at HMM, GMM and Gaussian levels:

- probabilities $P(X_{t,t+d}|H_i)$ of observation sequences $X_{t,t+d}$ given a HMM H_i are firstly computed during first pass of acoustic-phonetic decoding. Rescoring with upper-models requires the evaluation of $P(X_t|S_i)$, for each GMM S_i matching to the best phonetic sequence;
- as the search develops a part of the exploration graph, various concurrent hypotheses are evaluated. Each of them corresponds to a phonetic sequence. It is clear that word-hypotheses could share some phonetic subsequences competing
- state tying leads to involve the same state in computation of different HMM probabilities; the architecture of the acoustic handler should take advantage of this state sharing;
- Gaussian tying could lead states to share some of their Gaussian components. As Gaussian are involved into different mixtures, probabilities computation must be decoupled from the models structure at the state level.

2.4.2. Handler architecture

In order to avoid multiple computation of a likelihood, we separate clearly the search algorithm and the acoustic handler which is in charge of acoustic probabilities computing and caching. This handler is based on a 2-level caching mechanism; as the search algorithm has to score each hypothesis, it requires a probability $P(X_{t,t+d}|H_i)$.

The acoustic handler searches this score in the level-1 cache (L1); if it is not found, this score is computed by using the Viterbi algorithm and the probabilities of emission $P(X_t|S_i)$. These last ones are searched in the level-2 cache (L2). If not found, these values are computed by using a fast likelihood method which is described below. L1 and L2 caches are implemented as circular buffers. This last technique allows to decrease likelihood computing time by about 15%, compared to the classical FPU-based computation. Finally, likelihoods are computed on-demand, allowing to limit the computed scores to the ones effectively required by the search and to take benefit from the lexical and linguistic constraints.

The figure 2 describes the global architecture of the acoustic handler.

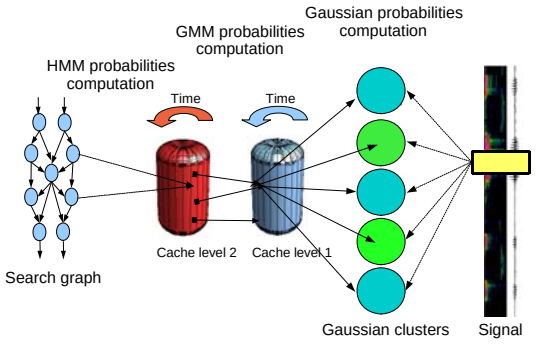

Fig. 2*. Architecture of Speeral acoustic handler; likelihood are computed on-demand, depending on the paths which are effectively developed by the search algorithm; the L1 cache stores acoustic probabilities of an observation sequence given an HMM; the L2 cache stores probabilities of an observation X_t for a state S_i . The fast likelihood estimation method is based on Gaussian clustering and dynamic selection of relevant clusters.*

On-demand strategy is supposed to limit the number of estimated probabilities. On the other hand, this strategy implies some additional operations related to cache management. In order to estimate the gain provided by this technique, we perform some tests where full and on-demand computation are compared. This comparison is

achieved on 4 acoustic models which include respectively 6000, 60000, 90000 and 230000 Gaussian components.

Results confirm that on-demand computation provides a significant gain in terms of decoding duration, which remains between 1.5 to 6xRT (corresponding respectively to 6k and 230K models). Nevertheless, it is clear that efficient acoustic handling does not allow to reach a decoding speed sufficient for spoken dialogue with computer, in spite of a strict pruning scheme and a very efficient likelihood computation (likelihood function is written in assembly language, using SIMD instruction set and 128 bits data alignment). This last technology provides an absolute speed-up of 15%, compared to classical FPU based function. On an Opteron 2.6GHz processor, we observe a computation speed of about 90.106 likelihood per second, for 39 coefficient feature vectors.

It is clear that models composed of more than 10 million parameters are not tractable under low-resource constraints without any fast computation method. We propose a method based on Gaussian selection (GS) which is described in the next section.

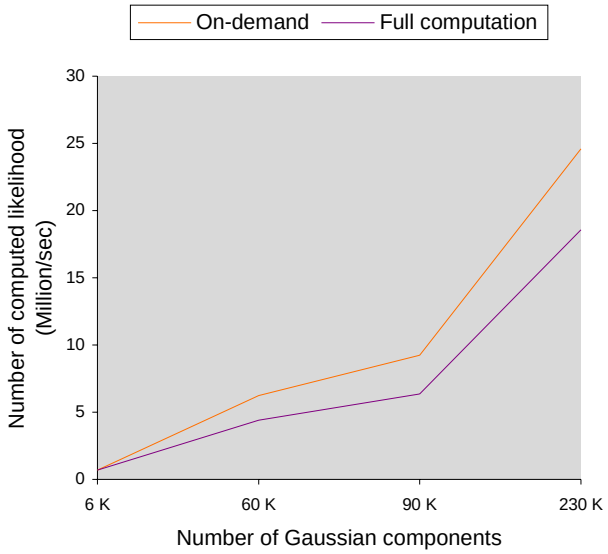

Fig. 3. *Impact of model complexity (in number of Gaussian components) on the number of computed likelihood. On demand strategy is compared to full likelihood calculation, according to the complexity of acoustic model. This method allows to reduce the computational cost of acoustic scoring; nevertheless, the number of computed likelihood grows linearly as the model complexity increases.*

2.5. Fixed-precision likelihood approximation

Numerous methods for fast likelihood computation have been evaluated the last decades. Most of them rely on Gaussian selection techniques which identify, in the full set of Gaussian, the ones which contribute significantly to the frame likelihood. We developed an original method which guaranties a constant precision ε of the likelihood approximation. This method consists in off-line clustering of Gaussian and in on-line selection of Gaussian clusters.

As proposed in some papers ([2],[4]), Gaussian are clustered by a classical k-means algorithm on the full set of Gaussian, using a minimum-likelihood loss distance. Each center of cluster is a mono-Gaussian model G_i resulting of the merge of all members of the cluster.

The on-line selection process consists in selecting a set of clusters which models the observation neighborhood. It is important to note that, on the contrary of classical Gaussian selection methods, the number of selected clusters is variable, according to the considered frame and to the expected precision ε .

The clusters are selected by computing the likelihood of the frame knowing each cluster center G_i; these likelihood are used for partitioning clusters into two subsets (tagged selected and unselected clusters) respecting the rules: (1) each frame likelihood knowing a selected cluster center is greater than each unselected one and (2) the sum of unselected clusters likelihood is lower than an a priory fixed precision threshold. Lastly, a posteriori probabilities are computed using only Gaussian belonging to selected clusters. Probabilities of unselected Gaussian are supposed to be close to zero; they are approximated by backing off to the cluster probabilities, which are computed using the Gaussian center of the related cluster.

This Fixed Precision Gaussian Selection (FPGS) method is designed to adapt dynamically the CPU time dedicated to acoustic processing according to an a priory fixed precision. In adverse acoustic conditions, it leads to remain a good acoustic precision limiting computational costs.

2.5.1. Evaluation

We test our method on the real-time configuration of our recognition engine, in the framework of ESTER evaluation campaign ([3]). ESTER aims to evaluate rich transcription systems on French broadcast news. Materials consist in 100 hours of radiophonic shows, recorded in various acoustic conditions.

Acoustic models are trained on ESTER materials. They includes 6000 HMM sharing 936 mixtures. State tying is performed by decision tree algorithm ([9]). HMM set contains globally about 60000 Gaussian components. In order to evaluate FPGS efficiency according to precision threshold and PDF cluster granularity, we estimated 2 set

of respectively 512 and 1200 clusters. The system involved in this experiment is based on a 27k words lexicon and full trigram language model.

The table 3 shows the impact of precision threshold both on the number of selected Gaussian and in terms of accuracy. Results are expressed according to the log-precision α with $\alpha = \log(1 - \varepsilon)$. This experiment is carried out on 1 hour of speech material extracted from ESTER broadcast news corpus. The granularity of the acoustic space partition seems to impact slightly the system performance. Results of table 3 and 4 suggest that larger set of Gaussian cluster is more efficient with high precision threshold, and worse in faster configuration. As expected, the Gaussian ratio for 1200-cluster configuration is lower than the one obtained with 512-cluster one: more precise partitioning of Gaussian set allows to target more precisely the components which have to be selected. On the other hand, the set of Gaussian center of cluster is twice bigger for the 1200-cluster scheme and more cluster probabilities are systematically computed. Moreover, in low precision configuration ($\varepsilon \leq -10$), additional cost due to greater Gaussian ratio is marginal, compared to the global cost of the decoding process. Therefore, observed gains of decoding speed remain low and finally, various clustering granularities seem to lead to very close performance.

2.6. Profiling recognition engine

In order to estimate the effective contribution of each component of recognition engine in the decoding duration, we performed profiling runs. The system is based on a 27k words lexicon and a fast pruning scheme. We tested 3 configurations based on acoustic models composed of respectively 60000 (60KG), 230000 (230KG) and 320000 (320KG) Gaussian components. Results (figure 4) show that, in spite of fast likelihood computation and caching mechanisms, acoustic model complexity impacts strongly on the decoding speed. With the smallest models, acoustic scoring takes about 20% of the decoding duration. Using smaller models brings only slight increase of decoding speed. Moreover, 320KG models impact negatively on the decoding speed without any improvement of accuracy, this last point being probably due to training conditions (especially the amount of training data).

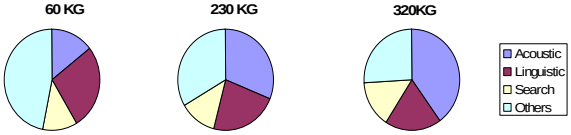

Fig. 4. *CPU-time requirements for the acoustic handler, lexical and linguistic access, search algorithm. Estimations performed on the 4xRT system based on a 27k words lexicon, and acoustic models composed by 60000(60KG), 230000(230KG) and 320000 (320KG) Gaussian components.*

3. LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

In this section, we report experiment results aiming to evaluate the impact of acoustic models and lexicon size on performance, both in terms of accuracy and resource consumption (memory footprint and CPU-time). Experiments are carried out in the framework of ESTER evaluation campaign, using a fast pruning scheme.

We tested 4 acoustic model profiles; the first is the one involved in the 10xRT systems (230KG). It contains 3600 mixtures of about 64 Gaussian each, corresponding to a total of 230000 Gaussian components. The second is a variant of 230Kg decreasing the number of components per mixture from 64 to 24, while the same topology remains. We obtain a model (90Kg) composed by 90000 components. The 3rd model (60Kg) is composed by 3000 HMM, 936 mixtures and 60000 components. Lastly, we use a very small context-independent models composed of 100 emitting states of 64 Gaussian each. All these models are gender dependent. Experiments are conducted with 2 dictionaries containing respectively 27k and 65k words.

Results are reported in table 5 for 27k lexicon, and table 6 for 65k lexicon. Reported memory footprint (*RAM occ*.) is evaluated on a speech segment of 22 seconds. It corresponds to the upper limit of memory used during the decoding run.

Table 5. *Real time factor (RT-factor), Word Error Rates (WER) and memory footprint (RAM occ.) of recognition system using 27k words lexicon. Test performed on 3 hours of ESTER development corpus*.

Acoustic model	6.5Kg	60Kg	90Kg
RT-factor	0.55	0.65	1.05
RAM occ.	201 M	227M	346M
WER	40.7%	28.5%	26.8%

Table 6. *Real time factor (RT-factor), Word Error Rates (WER) and memory footprint (RAM occ.) of recognition system using 65k words lexicon. Test performed on 3 hours of ESTER development corpus.*

Acoustic model	6.5Kg	60Kg	90Kg
RT-factor	0.6	0.95	1.25
RAM occ.	256 M	260M	380M

WER	40.3%	27.5%	25.6%

Results show that lexicon growing leads to increase decoding duration of about 20% to 30%, depending on the acoustic configuration, while the accuracy is slightly improved (about +1% absolute gain).

Considering the tradeoff between decoding speed and accuracy, context-free model seems not to be a good option: it allows to reach decoding speed close to the one obtained by 60Kg models, while the accuracy is dramatically worse (about 13% absolute WER!). It is clear that low precision models lead to increase paths to evaluate, spending in the search process the potential gain which was obtained by reducing the acoustic model complexity. Globally, system is able to quickly reach performance close to our 10xRT system, which is about 21% WER. Very fast configuration can be used, depending on resource availability and the targeted performance.

4. CONCLUSION

We present a design for fast LVCSR; the proposed methods for acoustic and linguistic models management has shown their ability in efficient handling of large vocabularies and high-resolution acoustic models: real time is reached by a system of 27k words and 90000 Gaussian components. The proposed methods preserve the functional model of the standard system; this approach allows to define very different configurations (from 0.5xRT to 10xRT) by changing macro-parameters such as pruning thresholds, acoustic resolution, acoustic and linguistic models, etc. Nevertheless, more efforts may be produced to embed this system in very light devices, where no floating point unit is available and in which the free memory remains lower than 100MB.

Finally, results show clearly that since good level of performance can be reached using a lower resource consumption level, optimal performances require very expensive modeling and exhaustive search graph exploration : our best results in unconstrained conditions are obtained by combining 2 systems, with a decoding time greater than 10xRT each, acoustic models of more than 200000 Gaussian and several millions of estimated trigrams or quadrigrams, for final performance close to 19% WER ([5]). It is clear that for custom application, ASR systems must be pruned according to the application needs (spoken dialogue system, speech mining, etc.) and the resource affectation has to be considered from client-oriented point of view.

5. REFERENCES

[1] Xavier L. Aubert. An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech and Language*, 16:89–114, 2002.

[2] E. Bocchieri. Vector quantization for the efficient computation of continuous density likelihood. In IEEE, editor, *Proc ICASSP'93*, volume 2, pages 692–696, Speech Research Dept.,AT&T Lab., Murray Hill, 1993. IEEE.

[3] S. Galliano, E. Geoffrois, D. Mostefa, K. Choukri, J.-F. Bonastre, and G. Gravier. The ESTER Phase II evaluation campaign for the rich transcription of French broadcast news. *In Proc. of the ECSCT*, 2005.

[4] K.M. Knill, M.J. Gales, and S.J. Young. Use of gaussian selection in large vocabulary continuous speech recognition using HMMS. *In Proc. ICSLP'96*, volume 1, pages 470–474, Philadelphia, PA, USA, 1996. Cambridge University.

[5] Benjamin Lecouteux, Georges Linarès, Yannick Estève and Julie Mauclair. System combination by driven decoding. *In ICASSP'07*, 2007.

[6] D. Massonié, P. Nocéra, and G. Linarès. Scalable language model look-ahead for LVCSR. *InterSpeech'05*, Lisboa, Portugal, 2005.

[7] Stefan Ortmanns, Andreas Eiden, and Hermann Ney. Improved lexical tree search for large vocabulary speech recognition. *In Proc. ICASSP*, Seattle, USA, May 1998.

[8] Stefan Ortmanns, Hermann Ney, and Andreas Eiden. Language- model look-ahead for large vocabulary speech recognition. In Proc. ICSLP, Philadelphia, USA, October 1996.

[9] S.H. Young and P.C. Woodland. State clustering in Hidden Markov Models-based continuous speech recognition. Computer Speech and Language, 8:369–383, 1994.