A DECODER FOR LARGE VOCABULARY CONTINUOUS SHORT MESSAGE DICTATION ON EMBEDDED DEVICES

Jesper Olsen, Yang Cao, Guohong Ding, Xinxing Yang

Nokia Research Center
Nokia House 1
No. 11, He Ping Li Dong Jie
Beijing 100013, P.R.China
{jesper.olsen, yang.1.cao, guohong.ding, xinxing.yang}@nokia.com

ABSTRACT

We present our recent progress towards implementing large vocabulary continuous SMS Dictation in embedded devices. The dictation engine we describe here is based on the popular finite state transducer paradigm and is capable of handling large vocabularies and high order n-gram language models in a small memory footprint - even relative to what is available in current high end devices such as the Nokia N800 Internet tablet and the N95 Symbian phone. We illustrate the performance of the engine on a 20k vocabulary Chinese Mandarin dictation task which requires less than 10Mb RAM memory to run on the device. The accuracy of the continuous engine is similar to the accuracy of the isolated word dictation engine we have previously developed.

Index Terms— Speech recognition, finite automata, mobile communication, text communication

1. INTRODUCTION

Large vocabulary speech dictation products have for a number of years been available for the desktop PC market, but in practice they have not been widely adopted by users. This has partly been due to technical issues, but also due to the fact that PCs are equipped with keyboards, which allow for efficient text input at a rate which – for an experienced typist – clearly exceeds the speaking rate. Mobile devices on the other hand are equipped with very limited keyboards, which complicates text input – particularly for languages such as Chinese where the number of input symbols is much higher than for English. Nevertheless, text messaging on mobiles is very popular even for these languages – In the Asia Pacific area alone, hundreds of billions of SMS messages are sent each year, and with the introduction of e-mail and internet enabled phones, the need for efficient mobile text input solutions is growing. This is the main motivation for us to bringing speech dictation into mobile phones, and we have previously described how we have implemented isolated word dictation for languages such as English and Mandarin Chinese by essentially augmenting an earlier isolated word "name dialer" ASR engine with a language model and a VAD module for pause detection [1]. In terms of memory and cpu usage, continuous speech dictation is much more demanding than isolated word dictation, but from the users point of view also more attractive because it allows for a more natural speaking style, and a higher text input rate.

2. EMBEDDED ENVIRONMENT

Cost, size and power consumption are factors that in practise limit the CPU and memory resources available on embedded platforms. Nevertheless, in recent years embedded platforms have become powerful enough to host even complex applications such as large vocabulary continuous speech recognition [2,4,5,6,8]. In this paper we describe our own effort to implement a large vocabulary speech recognition application targeted for embedded devices such as the Nokia N800 Internet tablet – a Linux device equipped with a 320 MHz ARM TI OMAP 2420 processor, 128 MB RAM, 256 MB flash ROM. The N800 has floating point support, which means that another traditional limitation of embedded applications has vanished: the need to implement all algorithms in fixed-point arithmetic.

3. ACOUSTIC MODELLING

Because of the scalability it affords, acoustic modeling in our engine is based on state tied triphone Hidden Markov Models (HMMs) [15]. Each triphone model has a left-to-right topology with 3 emitting states, and each state has 16 mixtures per state. The HMMs we currently use are within-word context dependent models – i.e. phone context is not modeled across word boundaries. Furthermore, transition probabilities between states are not used, because we have long found that they contribute very little to the modeling accuracy of the HMM set. In order to reduce the memory footprint and speed up mixture calculations, the models are subspace encoded [3]:

$$P(x) = \sum_{m=1}^{M} c_m \left(\prod_{k=1}^{K} N(x_k; \mu_{mk}, \sigma_{mk}^2)\right) \quad (1)$$

The means and variances are quantized within each subspace – quantization is not strictly required, but there would be no advantage in using the subspace representation without quantization. Subspace HMMs can be used with different subspace dimensionalities – if the feature space has dimension D, then at one extreme D 1-dimensional subspaces could be used, and at the other extreme 1 D-dimensional subspace could be used. For 1-dimensional subspaces we have found that 5-bit quantization (32 codebook elements) usually results in a quantization error that is so small that it has negligible influence on the recognition accuracy. Subspace quantization results in a significant memory footprint reduction because, instead of storing the mean and variance components directly as for instance 32-bit floats, we only need to store the 5-bit index to the subspace codebook – with each index referencing a Gaussian subspace mixture. The size of the codebook is typically insignificant – e.g. 39x32x2=2496 bytes for the common 39 dimensional "13 MFCC+delta+delta-delta" front-end. With this front-end, directly storing 16 39 dimensional mean/variance components (per HMM state) would require 16*39*4*2=4992 bytes. With bit-packing, the subspace representation only requires 16*39*5-bit=390 bytes – a saving of more than an order of magnitude. Additionally, the mixture likelihood calculation can be carried out significantly faster because the subspace pdfs can be pre-computed and looked up in a table each frame time. To further simplify the mixture calculations, we use the approximation:

$$P(x) = \max_{m=1}^{M} c_m \prod_{k=1}^{K} N(x_k; \mu_{mk}, \sigma_{mk}^2) \quad (2)$$

or in the log domain:

$$\log P(x) = \max_{m=1}^{M} \log(c_m) + \sum_{k=1}^{K} N(x_k; \mu_{mk}, \sigma_{mk}^2) \quad (3)$$

The max approximation is faster to compute because in addition to avoiding the need to add log domain probabilities, it also allows the calculation of the sum term in equation (3) to be broken off if/when the partial sum falls below the current maximum. Finally mixture weights are left out too because we have found that they have very little influence on the modeling accuracy:

$$\log P(x) = \max_{m=1}^{M} \sum_{k=1}^{K} N(x_k; \mu_{mk}, \sigma_{mk}^2)) \quad (4)$$

Leaving out the mixture weights has very little influence on speed, but there is a modest memory saving – depending on the number of mixtures in the model set. Table 1 below summarizes the character accuracy for the three different mixture calculation strategies corresponding respectively to equation 1 (baseline), 3 (max) and 4 (max and no weight). The results are from our internal SMS dictation test set which has a perplexity of around 100 (see table 2), and consists of about 9 hours of speech. The experiments have been repeated for three different HMM model sets with different number of mixtures – respectively 10k, 5k and 3k. For the 10k system, the recogniser based on (4) is roughly 25% faster than the baseline system. For the 5k, and 3k systems the gains are smaller (respectively 18% and 10%) due to mixture calculations taking less overall time in these systems. All accuracies in table 1, and in this paper, are based on speaker independent acoustic modelling.

	10k	5k	3k
sum & mix weights	81.40%	78.95%	75.62%
max & mix weights	81.39%	79.02%	75.87%
max & no mix weights	81.39%	79.02%	75.87%

Table 1: Character accuracy for three different mixture calculation strategies and 3 different HMM sets.

3. NETWORK SEARCH

Finite state transducer networks have in recent years become very popular in the speech recognition community [9,7,5]. Finite state transducers provide an elegant way of representing the different model components in a speech recogniser (HMMs, pronunciation dictionary, language model) and combining them into a minimal finite state network that can drive a speech recogniser. The language model we use in our dictation engine is based on n-gram modeling – in principle we use the same n-gram model as in our earlier isolated word dictation system [1], except that that particular system only made use of the 2-gram section – partly because of memory restrictions on the target platform it was developed for, and partly because reasonable word accuracies could be achieved with even a bigram LM in that system. Continuous speech recognition, however, is more challenging than isolated word dictation, and consequently it is more important to make use of the extra modeling accuracy that can be gained by using higher order n-grams.

The continuous dictation engine is based on a 1-pass architecture with a recognition transducer created on the basis of higher order n-grams (all n-gram sections are used –the network includes backoff paths from the highest level all the way down to 1-grams if necessary). An alternative to this would be to adopt a multi pass strategy similar to [6,5] where a bigram network is used in a first pass where a lattice is computed and then rescored with word trigram statistics in a second pass. The advantage of this is that a smaller recognition network can be used in the first pass. However, a 1-pass architecture where all knowledge sources are used in the first pass is in principle more accurate than a multi pass solution, because in a multi pass solution it is

usually impossible to recover from search errors in earlier passes. A multi pass architecture with rescoring is not difficult to implement on top of the current decoder, but we have not yet explored this possibility seriously. With the current models memory is not a major obstacle; the present engine requires less than 10MB RAM memory, which is not big problem on a device such as the N800. This could change, however, if we want to use significantly larger n-gram LMs in the recogniser, or if we want improve the acoustic modeling by using cross-word context dependent acoustic models.

3.1. LM Pruning

A problem with large vocabulary n-gram language models is that they require a lot of data for training, and that the size of the LMs tends to be proportional to the size of the corpora they have been trained on. An efficient way to deal with this is entropy pruning [13]. Table 2 below shows the number of n-grams in a 6-gram LM trained on a corpus of approximately 5 million Chinese words. Also shown is the perplexity of the LM for each n-gram order included. The statistics are shown for two different entropy pruning levels – light pruning (1e-10) and heavy pruning (8e-7). The perplexity figures indicate that above order 3, there is relatively little additional information in the n-grams. Table 3 shows the size of the recognition networks that results from these two LMs – the recognition networks are created on the basis of a 10k mixture HMM set. The heavily pruned network is based on all n-grams up to order 6, whereas the lightly pruned network is only based on the orders up to 3 (a 4^{th} order network could not be constructed in this case on the machine used for these experiments, because the intermediary transducer operations required too much memory). The heavily pruned network has almost 6 times fewer arcs than the lightly pruned network – although this comes at a cost of some accuracy. The static recognition network is clearly the largest component in the recognition network – the network is represented internally as a list of arcs grouped by start state. Additionally it is necessary for each node to store the index of the first arc belonging to that node in the arc list. Each arc stores the end node of the arc, the penalty of the arc and the transducer label (input, output) of the arc. How many bits are required depends on the complexity of the network (number of labels, quantization etc). For the networks described here, 7 bytes per arc are required – hence the large network requires 21.9Mb for storing arcs, and the small network only 3.7Mb. Since there are more than 2^{16} arcs, indexing the arc array requires 32-bits and hence the list of first-arcs is also a large data structure – requiring respectively 6.3Mb and 1.2Mb if stored as an array of 32-bit integers. However, half that amount can in practice be used by storing only a 16-bit index and dynamically calculating the 32-bit index by looking up the relevant offset for the individual nodes in a table.

Order	#n-grams & perplexity			
	E. Prun 1e-10		E.Prun 8e-7	
1	20002	379.24	20002	379.24
2	982029	105.04	157826	117.69
3	591050	84.40	46746	102.73
4	408867	81.46	9471	101.53
5	165733	81.30	430	101.50
6	55571	81.27	6	101.50

Table 2: 6-gram LM size for two different pruning thresholds.

Order	E Pruning	#nodes	#arcs	Acc
3	1e-10	1582429	3128598	83.16%
6	8e-7	291220	535418	81.39%

Table 3: Recognition network size and recognition accuracy.

3.2. Weight Quantization

It is well known that LM weights can be quantized to a very low level without significantly impacting recognition accuracy [14]. In our isolated dictation system, we have been able to quantize bigram probabilities using a 4-bit codebook – although that particular representation had the benefit of a linear interpolation scheme to reduce quantization errors [11]. Weights in a finite state transducer network cover a much wider range than the probabilities found in a particular n-gram section of a language model – this is so because the finite state network is constructed from several n-gram orders, as well as backoff probabilities, which all have different ranges. Furthermore, during the network construction, arc weights are combined and pushed around the network resulting in even more spread. Table 4 shows the character accuracies in an evaluation where respectively 32- and 8-bit quantized probabilities are used in combination with 3 different HMM sets with different numbers of mixtures. As can be seen, character accuracy is almost the same in the two sets of experiments. Using 8-bit quantized probabilities instead of the original 32-bit values saves 3 bytes per arc – corresponding to respectively 9.3Mb and 1.6Mb for the two recognition networks in table 3 above.

	10k	5k	3k
32-bit arc penalties	81.43%	79.02%	75.91%
8-bit arc penalties	81.39%	79.02%	75.87%

Table 2: Word accuracy with and without 8-bit quantization of arc penalties

3. BEAM PRUNING

Beam pruning is a simple way to make speed-accuracy trade offs in a Viterbi decoder. Figure 1 below shows the

accuracy–real time (RT) graph for three different LM & HMM combinations. The two first curves are for the 10k HMM set used in combination with the two LMs from table 2. These two curves do not intersect – at the same RT point the large LM is always more accurate. The third curve is for the small LM used in combination with the 5k model set – the 5k model set is less accurate than the 10k model set, but it is also significantly faster to use, and for stricter speed requirements, it is possible to achieve a better speed-accuracy tradeoff with this model set. By far the most time consuming part of the decoder is mixture calculations which take 60-80% of the decoder time for the LMs and model sets used in this paper. The RT scale in figure 1 is for our development environment – the embedded target environment is at least an order of magnitude slower.

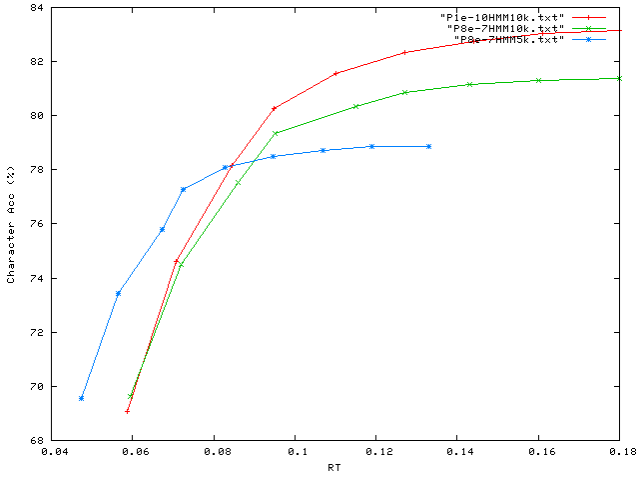

Figure 1: Accuracy versus Real Time (RT) for three different LM & HMM combinations.

5. CONCLUSION

We have presented a large vocabulary continuous recogniser we have developed for short message dictation on embedded devices. The character accuracy of the engine is comparable to what we have previously reported for isolated dictation [1]. Computationally, fast mixture pdf calculations is an important bottleneck in the decoder. Memory usage, on the other hand, is much less critical. In the experiments we reported here we did not include adaptation of acoustic and language models – in practice these are techniques that can significantly improve ASR performance on a personal device such as a mobile phone.

6. REFERENCES

[1] J. Alhonen, Y. Cao, G. Ding, Y. Liu, J. Olsen, X. Wang, X. Yang, "Mandarin Short Message Dictation on Symbian Series 60 Mobile Phones," *The Int. Conf. on Mobile Technology, Applications and Systems*, Singapore, 2007.

[2] E. Bocchieri and D. Blewett, "A Decoder for LVCSR Based on Fixed-Point Arithmetic", *Proc. of ICASSP, Toulouse* , pp. 1113-11116, 2006

[3] E. Bocchieri and B.K. Mak, "Subspace Distribution Clustering Hidden Markov Model", IEEE Transactions on Speech and Audio Processing, 9(3):264-275, March, 2001

[4] H. Franco, J. Zheng, J. Butzenberger, F. Cesari, M. Frandsen, J. Arnold, V. Ramana, R. Gadde, A. Stolcke, V. Abrash, "DynaSpeak: SRI's Scalable Speech Recognizer for Embedded and Mobile Systems", *Proc. of Human Language Technology Conference (HLT-2002)*, San Diego, USA, 2002

[5] I.L. Hetherington, "PocketSUMMIT: Small-Footprint Continuous Speech Recognition", *Proc of ICASSP,* Belgium, pp. 1465-1468, 2007

[6] D. Huggins-Daines et al, "POCKETSPHINX: A Free Real-Time Continuous Speech Recognition System for Hand-Held Devices", *Proc. of ICASSP,* Toulouse, pp. 185-188, 2006

[7] S. Kanthak, H. Ney, M. Riley and M. Mohri, "A Comparison of Two LVR search Optimization Techniques", *Proc. of ICSLP,* USA, pp. 1309-1312, 2002

[8] H. Kokubo, N. Kataoka, A. Lee, T. Kawahara, K. Shikano, "Embedded Julius: Continuous Speech Recognition Software for Microprocessor", *Int. Workshop on Multimedia Signal Processing*, Canada, October, 2006

[9] M. Mohri, F. Pereira and M. Riley, "Weighted Finite-State STransducers in Speech Recognition", *Computer Speech & Language*, 16(1):69-88, January 2002

[10] M. Novak, "Towards Large Vocabulary ASR on Embedded Platforms", *Proc. of Int. Conf. on Speech and Language Processing (ICSLP)*, Korea, 2004

[11] J. Olsen and D. Oria, "Profile Based Compression of N-gram Language Models", *Proc. of ICASSP*, Toulouse, pp. 1041-1044, 2006

[12] N.N. Schraudolf, "A Fast, Compact Approximation of the Exponential Function", *Neural Computation.* 1999;11(4):853-862

[13] A. Stolcke, "Entropy-based Pruning of Backoff Language Models", DARPA Broadcast News Transcription and Understanding Workshop, pp. 270-274, 1998

[14] E. Whittaker, B. Raj, "Quantization-based Language Model Compression", Proc. of Interspeech, Aalborg, pp. 33-36, 2001

[15] S.J. Young, J.J. Odell and P.C. Woodland, "Tree-Based State Tying for High Accuracy Modeling", *ARPA Workshop on Human Language Technology*, pp. 307-312, 1994