REDUCED LEXICON TREES FOR DECODING IN A MMI-CONNECTIONIST/HMM SPEECH RECOGNITION SYSTEM

Christoph Neukirchen, Daniel Willett, Gerhard Rigoll

Department of Computer Science
Faculty of Electrical Engineering
Gerhard-Mercator-University Duisburg, Germany
e-mail: {chn,willett,rigoll}@fb9-ti.uni-duisburg.de
www: www.fb9-ti.uni-duisburg.de

ABSTRACT

The presented work deals with the experimental identification of parts in a tree based decoder lexicon, that are more important for decoding efficiency compared to less important lexicon parts. Three different methods for constructing only the most important nodes in a set of tree lexicon copies are presented: building large trees; tree cutting; lexicon node removal. This leads to dramatic reduction of memory requirements while retaining the original recognition performance. In addition a reduction of the active decoding search space can be observed that leads to improved recognition speed. Although the presented methods can be generally applied to any HMM speech recognizer, experiments are performed in the hybrid MMI-connectionist/HMM system framework on the speaker independent 5k WSJ database.

1. INTRODUCTION

To improve decoding speed, tree organized lexicon structures are preferred to a linear lexicon in most HMM based large vocabulary speech recognition systems. When used in connection with bigram (or wide span) language models, a tree lexicon increases the memory requirements and a dynamic lexicon allocation strategy is needed. In this work we investigate the possibility to take the computational advantage of the tree lexicon structures without increasing the memory requirements by orders of magnitude. Thus a static lexicon representation in memory might be still applicable. For this purpose some techniques are applied that identify and remove those parts from the trees that seem to be less effective than other lexicon parts that have to be retained.

For the experiments these techniques are integrated in the decoder of the MMI-connectionist/HMM speech recognition system. In previous speaker independent large vocabulary continuous speech recognition tasks this hybrid system has been proven to be comparable to sophisticated continuous density HMM systems ([1], [2]). Although this system takes additional advantage from fast HMM likelihood calculation, the methods presented in the following can be applied to any HMM based speech recognition system.

2. SYSTEM DESCRIPTION

The MMI-connectionist/HMM speech recognition system consists of a combination of neural networks and discrete HMMs. The neural networks serve as vector quantizers (VQ) for mapping the continuous acoustic feature vectors on discrete VQ-labels. The neural vector quantizers are trained using a novel approach (presented in [2]) that maximizes the mutual information between the produced VQ-labels and the phonetic classes (here: (states of) HMMs). So the system avoids the inherent information loss that can be observed in classical discrete systems using k-means-VQs. The discrete HMMs incorporate State-of-the-art techniques like triphone modeling (word internal or cross-word) and parameter smoothing and reduction by a state-clustering method based on phonetic decision trees that has proven to be useful in LVCSR tasks. Since the hybrid system is based on discrete HMMs, local likelihood calculation can be done rapidly by table-lookup in the HMM states. This leads to a very fast decoding compared to standard Gaussian mixture approaches where calculation of the mixture densities usually takes more than 50% of CPU time during decoding [6]. In the MMI-connectionist/HMM system framework a two-pass decoding strategy is used: The first pass (that is described in this paper) incorporates only bigram language models and the triphone acoustic models in a time-synchronous beam search decoder. In a second pass wide-span language models (e.g. trigrams) can be applied by rescoring a word lattice produced in the first pass.

3. TREE LEXICON DECODING

3.1. Linear vs. tree lexicon

In small and medium vocabulary tasks (with a only a few up to 1000 different words) static linear lexicons are common in most speech recognition systems. For large vocabulary tasks, tree organized lexical representations are widely used because redundant computations in different words sharing the same beginning phones (or triphones) are avoided. This leads to a significant speed up due to less local likelihood calculations and less total probability updates. The effect of reducing the number of local (HMM) likelihood evaluations can be also obtained in a non-tree lexicon by simply caching all calculated HMM likelihoods. Typically the number of triphone instances in a large vocabulary tree lexicon is about a half smaller compared to a linear one. In a unigram (or zero-gram) language model case this leads to lower memory requirements and to a smaller search space. Due to the beam search (the most common pruning technique), that seems to be most effective in the first phonemes of a word (where the tree compression factor is extremely high), the observed improvements in decoding time are also even higher than a factor of two [5].

Since the identity of a specific word is unknown at the start of the tree lexicon, in a bigram (or a wider span language model) case one (or multiple) tree copies are necessary for each predecessor word. This leads to an increase of memory requirements, by a factor given by the number of different words, that makes the usage of a static lexicon representation prohibitive. Therefore a dynamic tree organization must be applied that ensures that only those tree copies that are actually needed are held in the system memory. This dynamic memory organization leads to an extra CPU load during speech decoding. In addition the usage of tree copies increases the search space because equal word paths in different copies of a tree have to be taken into consideration simultaneously. However, again due to the beam search pruning, the actual average number of tree copies needed for decoding is quite small.

In beam search decoders it seems to be useful to apply the language model information as soon as possible. When using linear lexicons the language model score can be applied before the acoustic score since each word identity is known when entering the lexicon. In a tree lexicon the full language model can only be applied when a word becomes unique, i.e. it is delayed compared to the linear case. To overcome this delay a language model look-ahead can be used that propagates the best language model score in each branch of a tree towards its root (see: [6]). This leads to the additional effect that a certain word (with high acoustic score, but low language model score) sharing initial phones with another word (with high LM score) will survive the beam search instead of being pruned in the linear lexicon case due to its low LM score. Thus in the case of a tree lexicon with language model look-ahead, the beam width can be set tighter.

3.2. Combining linear and tree lexicon

The usage of memory consuming tree copies can be avoided by different combinations of linear and tree lexicon representations. This allows for a static lexicon organization in memory. As shown in [3] and [7] a single lexicon tree can be used for that part of words in the language model that are modeled by the back-off bigram section, i.e. by unigrams, since the language model probabilities do not depend on the previous word. The rest of the words (with full bigram probabilities) are represented in a linear lexicon. This introduces a latent mistake in language model likelihood access because for full bigram words now the back-off component is also allowed. But since full bigram probabilities are usually higher than their back-off counterparts this has apparently minor effects in decoding. In [4] the former approach is extended by using a large tree (containing all words) for the unigram back-off part, and a set of smaller tree copies that contain only those words with full bigram probabilities. Each of these full bigram tree copies typically contains much fewer words compared to the unigram tree, since there are only few word pairs with full bigram probabilities.

The speech decoder used in the hybrid MMI-connectionist/HMM system generalizes these methods described above in a flexible way by allowing a large tree for the back-off part, small tree copies for some of the full-bigram words *and* linear representations for the rest of the full-bigram words simultaneously. In this decoder only the successor trees with the largest numbers of leafs (containing a large number of words) are constructed. Those remaining successor trees, containing a fewer number of words, are collapsed to a single linear lexicon. This leads to a reduction of memory requirements compared to a full set of tree copies while still using that part of tree copies that promises to be most effective because it contains the most words.

In addition, once a word becomes unique (i.e. the following phones in the word are not shared by another different word) in the back-off tree and in the successor trees those arcs to the following phones in all the trees are directly connected to the corresponding parts of the linear lexicon. Therefore in all the tree copies there will never appear a unique remaining part of any word. This reduces the number of nodes in the trees and improves decoding time since parallel calculations of identical words in different successor tree copies are collapsed to one single path of calculations in the linear lexicon. Since the full language model score has been take into account when a word in a tree becomes unique, no language model corrections must be applied when collapsing the unique word endings in the trees with the linear lexicon.

3.3. Tree cutting

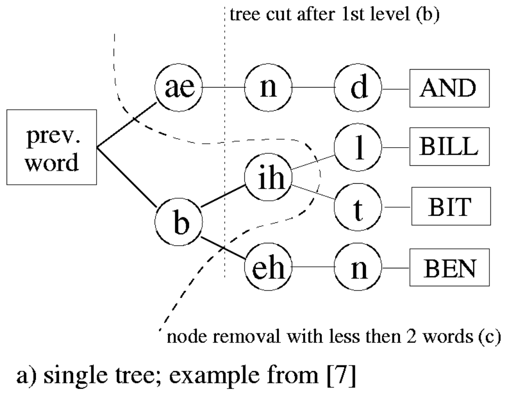
In [5] is shown that a new word under consideration during decoding is most ambiguous in the beginning phones and the decoder spends much effort in evaluating the first three phones. So the main advantage gained by using a tree lexicon representation is due to the first levels of the tree (where the tree compression factor is high). In the successive levels of the tree there are only very few branches and most of the words have become unique. This leads to the idea to use a tree lexicon representation only for the first phone levels of the tree by cutting off all the trees at a fixed depth. The arcs in the trees that are cut off are collapsed with the linear lexicon in a similar way as described in the previous section and shown in fig. 1b). At the cutting points, the remaining language model scores (that have not yet taken in account in the trees) must be applied for each word. This tree cutting method leads to smaller memory requirements since only the nodes in the beginning levels of the trees are needed in addition to the linear lexicon. Furthermore, the decoding effort may be reduced by collapsing parallel tree endings with the linear lexicon as described above. The method of cutting the trees at a fixed level can be combined with the one given in the previous section by building up and cutting only those trees that contain a certain amount of different words.

3.4. Removing ineffective tree nodes

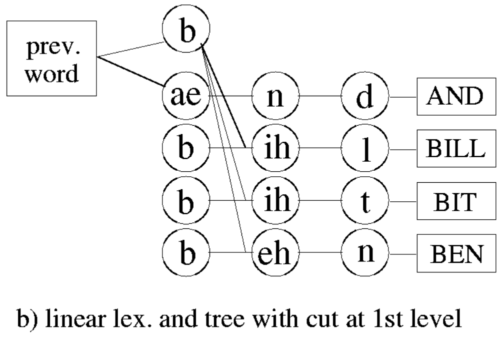
As explained above the tree structured lexicon is most effective in the first phone levels of the trees. This is due to the fact that many words share the same initial phones while only a few of them share the same phone sequence up near to the word endings. Thus the branching and compression factors near the tree roots are much higher than at the leafs. Since the advantage of the tree structure is more effective in phone nodes that are shared by many words compared to those nodes that are used only by a few words, it is worth to retain these useful nodes even if they are in a deep tree level (far away from the root). Cutting the tree at a fixed level near the root would destroy these useful nodes. On the other hand the phone nodes shared by a small number of words are less effective even if they are close to the tree root. Hence an alternative way to reduce memory requirements (without loosing the advantage of the tree structure) is to remove those nodes from the lexical trees that are shared by less than a fixed number of words. Again the removed tree nodes are collapsed with the linear lexicon (see fig. 1c)) and the missing language model scores are fully applied as in the cases described above. As in the case of cutting the trees above, usage of reduced trees can lead to less memory requirements due to node removal and to improved decoding speed because of collapsing equal words in different parallel successor trees. Also this method of reducing the tree structure by removal of less effective phone nodes can be combined with the methods of tree cutting and building large trees only.

4. EXPERIMENTS AND RESULTS

The parameters of the MMI-connectionist/HMM system are estimated using the SI84 training set of the WSJ0 corpus. The system uses 4 MMI-neural networks as vector quantizers with an output layer size of 300 for each VQ. The NNs quantize the following acoustic feature vectors: 12 MFCCs (incl. mean removal), 12 Δs, 12 $\Delta\Delta$s and logPower+Δ+$\Delta\Delta$. There are 11000 HMMs as word internal triphone models that are state-clustered using a phonetic decision tree. All tests performed in the experiments use the 5k Nov. 92 evaluation test sets with 330 sentences of 8 different speakers and the standard 5k ARPA



tree cut after 1st level (b)
prev. word
ae
n
d
AND
l
BILL
ih
t
BIT
b
eh
n
BEN
node removal with less then 2 words (c)
a) single tree; example from [7]



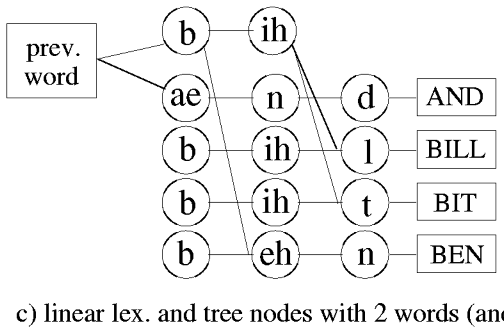


Figure 1. Reducing a lexicon tree: a) tree cut after 1st level; b) removal of nodes that contain less than 2 words

bigram language model. The pronunciations are taken from the lexicon provided by CMU ([8]) that gives us 50 different phones plus silence and an optional inter-word silence.

The recognition results for various acoustic beam width settings are shown in table 1. Although the system allows the application of a distinct word-end beam width and of a histogram pruning that limits the number of active models these are not used in the experiments presented here. All tests were performed on a DEC ALPHA XL366 workstation. The initial experiments shown in the first row as in [1] make use of a 5k linear lexicon that uses more than 46 thousand static triphones nodes (incl. silences).

When the complete set of 5852 successor tree lexicon copies is built up (with collapsing the unique word endings in all trees to the linear lexicon) the total number of triphone nodes increases by a factor of 20 (more than 870 thousand nodes) as shown in the second row of table 1. This makes the usage of static lexicon representations difficult due to high memory requirements. On the other hand the decoding speed is increased by a factor of 3 at similar recognition rates. Furthermore it can be seen that the acoustic beam width can be set much tighter compared to the linear lexicon at similar recognition rates.

The third row of table 1 shows the results for building just two tree copies, one for the back-off tree and one for the tree of sentence starting words. It can be seen that such system only uses slightly more (55 thousand) static triphone nodes compared to the linear case, but recognition rates and decoding speed is degraded compared to the full tree copy system.

In the following experiments (no. 4–9) only those tree copies are established that contain more than 5000, 4000, 3000, 2000, 1000 or 500 words resulting in 4, 15, 32, 66, 186 or 549 different successor trees. The remaining trees containing less words are collapsed to the linear lexicon as well as the unique parts in the trees. This increases the number of triphone nodes up to the half number of a full set of tree copies, but the decoding speed becomes even slightly better at comparable recognition rates due to avoiding calculations in trees without many words. That indicates it is only worth building up those trees that contain a certain number of words.

The experiments described from the 10th up to the 16th row make use of a set of 549 tree copies containing more than 500 words. These trees are cut off after a fixed depth level of triphone nodes. Cutting reduces the total number of triphone nodes in the tree copies by approximately a half if the trees are cut after the first triphone nodes. This reduction is also observed in the case of building up and cutting the full set of tree copies (Experiment no. 17). Table 1 indicates that neither recognition rates nor decoding speed is degraded by cutting. That means the main advantage of a tree lexicon is gained by the first level of triphone nodes.

In the experiments shown in the 18th – 27th rows of table 1 removal of nodes that are shared by only a few words is applied. First the set of 549 tree copies containing more than 500 words is constructed. Then, all triphone nodes in the trees that are shared by less than given minimum number of words are merged with the linear lexicon. In the best cases of these experiments only about 89 thousand nodes are totally needed although recognition rates are still only slightly degraded and decoding speed is even higher compared to the full tree copy system. In the last three experiments the full set of tree copies is reduced by the inefficient (i.e. sparsely shared) triphone nodes. This gives a system with high decoding speed at the best recognition rates with less than four times higher memory requirements compared to a linear lexicon. This still allows for a simple static representation of a triphone lexicon in computer memory using reduced trees.

Exp.	Lexicon	# trees	max.	min.	# triph.	beam=70		beam=80		beam=100		beam=120	
No.	type		depth	follwrs	nodes	WER	RT	WER	RT	WER	RT	WER	RT
1	linear	0			46750			16.8%	3.5	11.5%	8.9	10.6%	15.8
2	full trees	5852			872518	16.2%	1.0	12.7%	2.2	10.7%	5.1		
3	full trees	2			55124			13.3%	3.0	10.8%	7.1	10.5%	13.5
4	full trees	4			64471			13.5%	2.7	10.8%	6.6	10.5%	12.5
5	full trees	15			103976			12.8%	2.2	10.7%	5.7	10.5%	12.0
6	full trees	32			148217			12.8%	2.0	10.7%	5.4	10.5%	11.8
7	full trees	66			213407			12.7%	1.9	10.7%	5.1	10.5%	11.6
8	full trees	186			335512	16.2%	1.0	12.7%	1.8	10.7%	4.8	10.5%	11.3
9	full trees	549			499264			12.7%	1.8	10.7%	4.7	10.5%	11.3
10	cut trees	549	7		494057			12.7%	1.8	10.7%	4.7	10.5%	11.4
11	cut trees	549	6		487877			12.7%	1.8	10.7%	4.8	10.5%	11.4
12	cut trees	549	5		476501			12.7%	1.7	10.7%	4.7	10.5%	11.3
13	cut trees	549	4		455845			12.7%	1.8	10.7%	4.7	10.5%	11.3
14	cut trees	549	3		415908			12.7%	1.7	10.7%	4.7	10.5%	11.3
15	cut trees	549	2		353116	16.2%	0.9	12.7%	1.7	10.7%	4.7	10.5%	11.4
16	cut trees	549	1		265218	16.4%	0.9	12.8%	1.8	10.7%	4.7	10.5%	11.1
17	cut trees	5852	1		516804	16.3%	0.8	12.8%	1.6	10.7%	4.6	10.5%	11.3
18	red. trees	549		2	223172	16.4%	0.9	12.9%	1.6	10.8%	4.4	10.5%	10.1
19	red. trees	549		3	145428			12.9%	1.7	10.8%	4.7	10.5%	10.1
20	red. trees	549		4	109035			13.0%	1.7	10.8%	4.5	10.5%	9.9
21	red. trees	549		5	89461			13.1%	1.8	10.8%	4.7	10.5%	9.9
22	red. trees	549		6	77924			13.1%	1.8	10.9%	4.8	10.5%	9.9
23	red. trees	549		7	70560			13.1%	1.9	11.0%	4.8	10.5%	10.0
24	red. trees	549		8	65713			13.5%	1.9	11.0%	5.0	10.5%	10.1
25	red. trees	5852		2	319566	16.4%	0.9	12.8%	1.7	10.8%	4.7	10.5%	10.1
26	red. trees	5852		3	187660	16.4%	0.9	12.9%	1.8	10.8%	4.5	10.5%	9.8
27	red. trees	5852	3	3	183661	16.4%	0.8	12.9%	1.6	10.8%	4.5	10.5%	9.7

Table 1. WSJ Nov. 92 5k decoding error rates and real-time factors for different static lexicon representations

5. REMARKS

All the experiments presented here make use of a static lexicon representation in memory. Since the introduced reduced tree copies are used in addition to a linear lexicon, the minimum memory requirements are bounded by the requirements for a linear lexicon. This means in cases where a static representation of a linear lexicon becomes impossible (very large vocabulary, wide-span language models), adding reduced static trees will not work. But the improvements in decoding speed obtained by reducing the tree copies may be also gained in systems that make use of a (partly built up) dynamic tree lexicon representation.

6. CONCLUSIONS

We have presented three different ways for reducing the static memory requirements of set of tree structured lexicon copies in a decoder: i) building up only large trees; ii) tree cutting at a fixed depth; iii) removal of sparsely used tree nodes. These methods can be used in combination without degrading recognition performance while reducing memory allocation by factors of five to ten. Moreover these methods additionally seem to lead to a smaller search space resulting in improvements of decoding speed compared to the full set of tree copies. Although all experiments shown here used the MMI-connectionist/HMM system, tree reduction can be applied to any HMM decoder.

7. ACKNOWLEDGMENTS

The authors would thank to Jörg Rottland from Gerhard-Mercator-University, Duisburg for help with the WSJ database and for training the acoustic models. This work was partly supported by the DFG (German Research Foundation) under contract Ri 658/3-1. Responsibility for the content of this paper is with the authors.

8. REFERENCES

[1] J. Rottland, Ch. Neukirchen, D. Willett, "Performance of hybrid MMI-Connectionist/HMM ystems on the WSJ database, *Proc. IEEE-ICASSP,* 1997, pp. 1747–1750.

[2] Ch. Neukirchen, G. Rigoll, "Advanced training methods and new network topologies for hybrid MMI-Connectionist/HMM speech recognition systems", *Proc. IEEE-ICASSP,* 1997, pp. 3257–3260.

[3] H. Murveit, P. Monaco, V. Digalakis, J. Butzberger, "Techniques to achieve an accurate real-time large-vocabulary speech recognition system", *Proc. ARPA HLT Workshop,* 1994, pp. 393–398.

[4] G. Antoniol, F. Brugnara, M. Cettolo, M. Federico, "Language model representation for beam-search decoding", *Proc. IEEE-ICASSP,* 1995, pp. 588–591.

[5] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder, "Improvement in beam search for 10000-word continuous speech recognition", *Proc. IEEE-ICASSP,* 1992, pp. 9–12.

[6] S. Ortmanns, H. Ney, A. Eiden, "Language-model look-ahead for large-vocabulary speech recognition," *Proc. ICSLP,* 1996, pp. 2095–2098.

[7] J.J. Odell, "The use of context in large vocabulary speech recognition," *PhD thesis Cambridge University,* 1995.

[8] ftp: ftp.cs.cmu.edu/project/fgdata/dict/cmudict.0.4.Z