EFFICIENT LANGUAGE MODEL LOOK-AHEAD PROBABILITIES GENERATION USING LOWER ORDER LM LOOK-AHEAD INFORMATION

Langzhou Chen and K. K. Chin

Toshiba Research Europe Limited, Cambridge Research Lab, Cambridge, UK
{langzhou.chen,kkchin }@crl.toshiba.co.uk

ABSTRACT

In this paper, an efficient method for language model look-ahead probability generation is presented. Traditional methods generate language model look-ahead (LMLA) probabilities for each node in the LMLA tree recursively in a bottom to up manner. The new method presented in this paper makes use of the sparseness of the n-gram model and starts the process of generating an n-gram LMLA tree from a backoff LMLA tree. Only a small number of nodes are updated with explicitly estimated LM probabilities. This speeds up the bigram and trigram LMLA tree generation by a factor of 3 and 12 respectively.

Index: language model, Speech Recognition, decoding

1. INTRODUCTION

Language model look-ahead (LMLA) can accelerate the n-gram decoding process [1]. The basic idea of LMLA is to use look-ahead probabilities as linguistic scores when the current word is unknown. This leads to more efficient pruning in the decoding process, as the linguistic score is used before the end of the word is detected. Using a higher order LMLA will generally improve pruning efficiency because of the use of a more accurate linguistic score. However, due to the high calculation cost of generating the LMLA probabilities using a LMLA beyond bigram poses some difficulty. When a high order LMLA, for example trigram LMLA is adopted, the number of different trigram contexts that occur in the search space increases dramatically compared to the number of bigram contexts. As a result, the cost of generating trigram LMLA trees outweighs the benefit of more efficient pruning. Various method have been proposed to overcome this problem, including node based LMLA probability cache, pre-calculating the LM probabilities and perfect hashing [2]. Most of these methods focus on how to cache and look up the LMLA probabilities efficiently. However, generating the LMLA probability itself is still a very time consuming process. This paper focuses on efficient LMLA probability generation. A new method to generate the higher order LMLA probabilities from the lower order LMLA trees is presented. The method takes advantage of the sparseness of the n-gram LM to avoid unnecessary computation. In a backoff based LM, given the word context information, only a small part of n-gram probabilities are estimated explicitly, while the rest are backoff estimates. This fact lead to the development of a LMLA probabilities generation algorithm that achieves its goal by updating a (n-1)-gram LMLA tree. Only nodes that are related to explicitly estimated n-gram value are updated, the rest of the nodes are backoff of the corresponding nodes in the (n-1)-gram LMLA tree. A trigram LMLA system based on this new algorithm has been implemented in the Toshiba LVCSR system and results in significant CPU cost reduction.

The rest of this paper is organized as follows: in section 2, the calculation cost of the LMLA is analyzed, in section 3, a new efficient method for calculating the LMLA probability is introduced. A multi-layer cache structure to construct the trigram LMLA is presented in section 4. Finally, experimental results and conclusion are presented.

2. THE COMPUTATION COST OF LMLA

During the decoding process, the LM look-ahead probabilities have to be generated on-line. Therefore, the efficiency of calculating the LMLA probabilities has a direct impact on the speed of decoding. In the traditional method a dynamic programming process is used to calculate the LM look-ahead probabilities.

Given a particular LM context, the calculation of LM look-ahead probabilities can be divided into 2 parts. The first part calculates the LM probabilities of every word in the vocabulary based on the LM context. The second part involves assigning LM look-ahead probability to every node in the LM look-ahead network through a dynamic programming procedure. Supposing that the vocabulary contains V words and the LM look-ahead network contains M nodes. This means that for each LM history occurring in the search space, the LVCSR system has to look up V probabilities in step 1 and generate M look-ahead probabilities in step 2. Figure 1 shows the calculation of LMLA probability based on this method.

The values of V and M are quite big in LVCSR system. Typically during the recognition process of one sentence,

there are several hundred bigram contexts and several thousand trigram contexts occurring in the search space. For the higher order n gram, for example, four gram, the number of LM contexts in the search space is even bigger. For every LM context, the LM probability calculation mentioned above has to be carried out. Calculating LM look-ahead probabilities can take up significant CPU time.

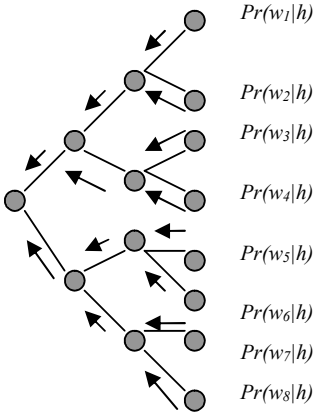

Figure 1 generating LMLA probability based on traditional method

3. THE NEW METHOD OF EFFICIENT LMLA PROBABILITY GENERATION

The traditional method treats the calculation of the LM probabilities and the LMLA probabilities as two independent processes. However, the calculation of LMLA probabilities is strongly related to the calculation of the LM probabilities. Making use of this relationship can results in significant CPU cost reduction.

3.1. The data sparseness of n-gram model

The n-gram language model uses the most recent n-1 history word to predict the probability of the current word. The backoff based n-gram LM can be expressed as:

$$P(w|h)=\begin{cases} f(w|h) & if(C(h,w)>0) \\ f(w|h')*Backoff(h) & otherwise \end{cases} \quad (1)$$

where *f(.)* is the discounted LM probability read from n-gram file, *C(.)* is the frequency of the event occurring in training corpus, *Backoff(h)* is the backoff parameters of history *h,* and *h'* represents the lower order history of *h*. Eqn 1 indicates that when the history-word pair can not be found in the n-gram data, the lower order model is used as the back-off estimate. Practically speaking, for large vocabulary applications, given a history *h,* the number of different history-word pairs that can be found in the training data is much smaller than the size of the vocabulary *V*. This means that for every word history *h*, most of the n-gram probabilities are given by the back-off estimate. This phenomenon can be used to accelerate the calculation in language modeling. For example, in [3], it is used for efficiently calculating the relative entropy when n-gram parameter is pruned. In [4], it is used to calculate the normalization parameters for MDI estimation. In this paper, it is used for efficient LMLA probabilities generation.

3.2. Calculating the LMLA probabilities from lower order LMLA information

.

The method presented in this paper uses the low-order LM look-ahead information to reduce the calculation cost of the higher order LM look-ahead. In this new method, only the LM look-ahead probabilities in a small subset of the nodes need to be updated, while for most of the nodes in the LM look-ahead tree, their LM look-ahead probability can be copied directly from the backoff LM look-ahead tree. The definition of the LM look-ahead in node *n* is the maximum LM probability over all the words that can be reached from *n*, which can be expressed as:

$$\pi(n|h)=\max_{w\in W(n)} \mathrm{P}(w|h) \quad (2)$$

where *W(n)* represents the set of the words that can be reached from node *n*. According to Eqn2, the definition of LM look-ahead can be re-written as:

$$\pi(n|h)=\max\{\pi_1(n|h),\pi_2(n|h)\} \quad (3)$$

where

$$\pi_1(n|h)=\max_{w\in W(n)\&\&C(w,h)>0} f(w|h) \quad (4)$$

and

$$\pi_2(n|h)=\max_{w\in W(n)\&\&C(w,h)=0} f(w|h')*backoff(h) \quad (5)$$

Therefore, the nodes in the LMLA tree can be divided into 2 parts, i.e.

$$\begin{aligned} N &= N_1 \bigcup N_2 \\ N_1 &= \{n|\pi(n|h)=\pi_1(n|h)\} \\ N_2 &= \{n|\pi(n|h)=\pi_2(n|h)\} \end{aligned} \quad (6)$$

Given a word history *h*, only the LMLA probabilities which are related to N_1 need to be calculated using the explicit n-gram estimates, while the rest of LMLA probabilities which are related to N_2 are calculated using backoff estimates.

Based on the analysis given above, a new method for calculating the LMLA probability is presented. The new method can be divided into 4 steps.

Step 1: generating a lower order LM look-ahead network, *T*, for each node *n* in *T*

$$\pi(n \mid h') = \max_{w \in W(n)} \Pr(w \mid h') \quad (7)$$

Step 2: multiplying lower order LM look-ahead probabilities by the backoff parameters of history *h*, to generate a backoff LM look-ahead network tree, *T'*, for each node *n* in *T'*

$$\pi'(n \mid h) = backoff(h) * \max_{w \in W(n)} \Pr(w \mid h') \quad (8)$$

Step 3: for each word *w* that co-occurred with the LM context *h* in the training corpus, replace the backoff LM probability in the leaf nodes of *T'* with the explicit LM probabilities in n-gram model, i.e. if *C(h,w)>0,* using *f(w|h)* to replace *f(w|h')*backoff(h)* in *T'*.

Step 4: for each word *w* in *W={w|C(h,w)>0}*, update the LM look-ahead probabilities in the node from which the *w* can be reached, using the dynamic programming procedure.

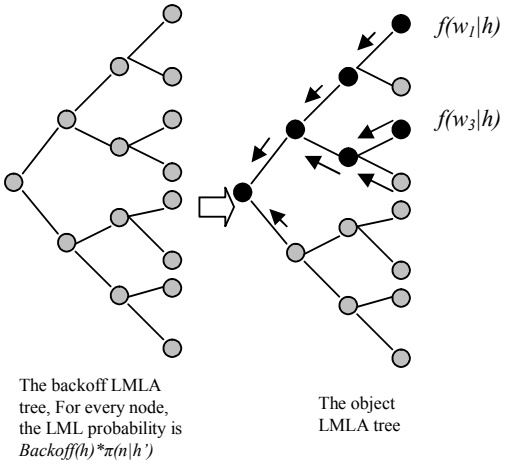

Figure 2 generating the LMLA probability using the new method

Figure 2 shows the calculation of LMLA probabilities based on the new method. The new method to calculate the LMLA probabilities starts from a backoff LMLA tree. The LMLA tree in **Figure 2** contains 8 leaves, i.e. 8 individual words. Given the LM context h, supposing that only 2 words: w_1 and w_3, have explicit LM probabilities, the new method only needs to calculate the LMLA probabilities in the nodes from which the w_1 and w_3, can be reached, i.e. the black nodes in **Figure 2**, while the rest of the LMLA probabilities, i.e. the LMLA probabilities in the grey nodes, can be copied from the backoff LMLA tree directly. The proposed method reduces the CPU cost significantly by calculating only a subset of nodes in the LM look-ahead tree, i.e. the nodes belong to N_1 in Eqn 6, instead of updating every node like the old method. For a particular LM context *h*, the word set *W={w|C(h,w)>0}* is much smaller than the whole recognition vocabulary. Nodes in N_1 are only a subset of nodes in LMLA tree.

4. MULTI-LAYER CACHE SYSTEM FOR LMLA

In the proposed method, the calculation of higher order LMLA probability depends on the value of the lower order LMLA probabilities, a multi-layer cache structure facilitates the generation of a high order LMLA. The number of the of layers is the same as the order of LMLA adopted by the decoder. There is a cache for the LMLA probabilities of each order. If a requested LMLA tree does not exist in the cache, it is generated using the corresponding backoff tree from low order LMLA cache.

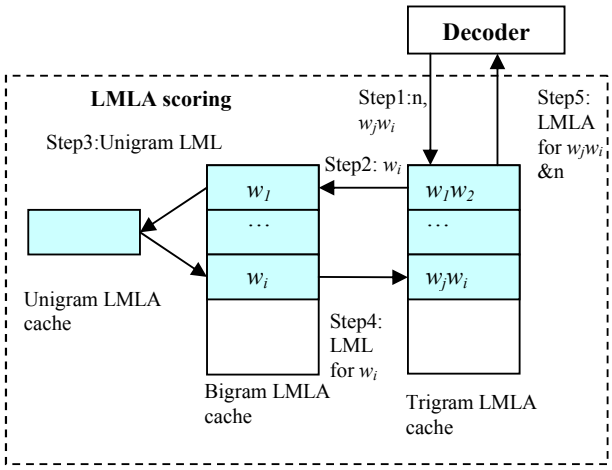

Figure 3 trigram LMLA based on multi-layer cache structure

Figure 3 shows the framework of trigram LMLA based on the multi-layer cache. Based on this structure, trigram LMLA can be divided into 5 steps:

Step 1: the decoder requests the LMLA score for node *n* and word history w_jw_i from the LMLA scoring module. The LMLA scoring module checks if the requested LMLA probability is already in the trigram LMLA cache, if it is, then go to step 5, otherwise, go to step 2.

Step 2: the bigram word history w_i is used to look-up the bigram LMLA cache, if the LMLA buffer for w_i is already in the cache, go to step 4, otherwise go to step 3.

Step 3: using the unigram LMLA buffer, the bigram LMLA buffer for w_i is generated and cached.

Step 4: using the bigram LMLA buffer for w_i, the trigram LMLA buffer for w_jw_i is generated and cached.

Step 5: the requested LMLA probability is returned to the decoder.

5. EXPERIMENTAL RESULTS

The experiments of trigram LMLA in this paper is based on the WSJ 20k vocabulary system. The training speech is WSJ0 and WSJ1 corpus, SI284 training set. The vocabulary is a closed vocabulary with about 20k words provided by Linguistic Data Consortium (LDC). The acoustic model contains 6000 tied HMM states with 10 Gaussian mixture

components per state. The speech vector is 33 dimensional with 10 proprietory cepstral features, 1 LOG energy, and their first-order and second-order time derivatives. The LM is a trigram language model trained by WSJ87-89 text corpus with about 40M words.

Configuration of LMLA	Decoding cost (MIPS)	CPU cost for LMLA
Taditional bigram LMLA	6054	25.8%
Bigram LMLA with new method	5328	7.5%
Traditional trigram LMLA	9589	65.8%
Trigram LMLA with new method	5280	10.0%

Table 1 The CPU cost comparison for LMLA between traditional method and the new method.

Table 1 shows the CPU cost of LMLA with different configuration based on the same beam width. It can be seen that the method proposed in this paper greatly reduced the calculation cost of LMLA probabilities. The process of generating the bigram and trigram LMLA tree is accelerated by a factor of 3 and 12 respectively. When trigram LMLA is used, the number of different trigram contexts occurring in the search space is much more than the number of bigram context. In the old method, the benefit of trigram LMLA can not compensate the extra calculation brought by LMLA and the system is even slower than the bigram LMLA system. On the other hand, because the new method calculates the LMLA probability much faster than traditional method, trigram LMLA speed-up the system further compared to the bigram LMLA, when the new method is used.

Table 2 shows the performance of the trigram LMLA and the bigram LMLA based on the new method. To achieve the same WER, the decoding based on trigram LMLA is always faster than the decoding with bigram LMLA. Trigram is more efficient in fast decoding, when the Beam width is 160, the WER of trigram LMLA is 1% better than bigram LMLA, when the beam width increased to 200, the difference reduces to 0.3%.

As mentioned in section 4, trigram LMLA in the new method is based on a multi-cache structure in which the trigram LMLA probabilities is generated from bigram LMLA probabilities and the bigram LMLA probabilities is generated from unigram LMLA probabilities. **Table 3** shows the calculation quantities of different order of LMLA based on one of test utterance in WSJ 20k task. It can be seen that even there are three times more trigram LMLA trees generated, it only occupies 1.26% CPU cost, while the bigram LMLA occupies 7.63% CPU cost. This is mainly due to the sparseness of the trigram data. Because the trigram data is very sparse compared to bigram data, the nodes to be updated in trigram LMLA are much less than those in bigram LMLA. Therefore, most of the calculation cost is from bigram LMLA even it is called less frequently.

Beam width	Bigram LMLA		Trigram LMLA	
	MIPS	WER	MIPS	WER
160	2490	13.9%	2476	12.9%
170	2975	11.4%	2959	11.0%
180	3604	10.2%	3572	9.9%
190	4378	9.5%	4329	9.0%
200	5328	9.2%	5280	8.9%

Table 2 The comparison results of trigram LMLA and bigram LMLA based on new method

	# of LMLA trees be generated	The CPU cost for LMLA
Bigram LMLA	322	7.63%
Trigram LMLA	1247	1.26%

Table 3. calculation quantity of different orders of LMLA based on new method

6. CONCLUSIOTN

In this paper, an efficient method to speed-up the calculation of the LMLA is presented. Contrary to the traditional method which treats the LM scoring in the leaves of LMLA tree and the LMLA probabilities propagation in the nodes as two independent processes, the new algorithm makes use of the sparseness of the n-gram data and calculates the higher order LMLA probabilities from the lower order LMLA probabilities. The method proposed in this paper is significantly faster than the traditional method. When a high order LMLA, such as trigram LMLA is adopted, the extra calculation cost is tiny compared to the bigram LMLA.

7. REFERENCES

[1] S. Ortmanns, H. Ney, and N. Coenen, "Language model Look-ahead for large vocabulary speech recognition," *In Proceedings of ICSLP*, Piladelphia, USA, pp. 2095-2098, 1996.

[2] A. Cardenal-Lepez, F.J. Dieguez-Tirado and C. Garcia-Mateo, "Fast LM look-ahead for large vocabulary continuous speech recognition using perfect hashing," *In Proceedings of ICASSP,* Orlando, USA, pp. I-705-I-708, 2002

[3] A. Stolcke, "Entropy-based Pruning of Backoff Language models," *In Proceedings of DARPA broadcast news transcription and understanding workshop*, Lansdowne, pp 270-274, 1998

[4] M. Federico, "Efficient Language Model Adaptation through MDI Estimation," *In Proceedings of EUROSPEECH*, Budapest, pp. 1583-1586, 1999.