AN EFFICIENT ONE-PASS DECODER FOR FINNISH LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Janne Pylkkönen
Neural Networks Research Centre, Helsinki University of Technology, Finland

Abstract

This paper describes a design of a one-pass large vocabulary continuous speech recognition decoder aimed for Finnish. The decoder is based on the popular time-synchronous beam search approach, extended to handle some language dependent issues. For the construction of the static part of the search network a new algorithm is presented, which enables efficient use of shared state HMMs. The search network also includes statically expanded cross-word triphone contexts, which increase the memory requirements only modestly. The beam search is enhanced with a bigram language model look-ahead technique, implemented using simple table lookups and an efficient caching scheme. Compared to our previous decoder, the new design achieves 24% relative reduction to phoneme error rate, with a near real-time performance.

Keywords: static search network, cross-word triphone contexts, language model lookahead

1. Introduction

The continuing increase in the complexity of acoustic and language models used in speech recognition has imposed growing requirements to the efficiency of decoders. Especially the introduction of cross-word acoustic models and long-span language models has led to development of many new solutions for the decoding problem. Compared to many established solutions in the field of speech recognition, the selection of different decoding techniques is relatively broad. A good overview of these is given in (Aubert 2002).

To promote the simplicity of design, we have chosen to use the one-pass time-synchronous approach. The problem is then how to build an efficient search network, and which knowledge sources it should contain. Probably the most fine-tuned solutions for the search network optimization are the WFST methods (Mohri et al. 2002). The idea is to combine all the knowledge sources together statically, and optimize the network for maximal efficiency. This, however, can restrict the complexity of the knowledge sources, and prevent some on-the-fly adaptation. A completely opposite solution is to expand the search network dynamically (Sixtus and Ney 2002). This, on the other hand, may be computationally too expensive for efficient decoding.

The architecture adopted for our decoder is in between these two network expansion methods. Similar to the approach presented in (Demuynck et al. 2000), the lexicon and acoustic models are combined into a static network, and the language model is used dynamically during the search. In this paper, we present the search network developed for our decoder, and also a novel language model lookahead scheme used to enhance the beam search. The target language has also imposed requirements to the decoder. Some problems in Finnish large vocabulary speech recognition (LVCSR) are highlighted and solutions to them are presented.

2. Language dependent issues

Finnish is a highly inflectional compounding language, so the number of distinct word forms in everyday use is very large. It is therefore very difficult to construct a good n-gram model based on words as lexical units. Instead of words, we use morpheme-like sub-word units called morphs which are discovered from a large corpus in an unsupervised manner (Siivola et al. 2003). The size of the lexicon can then be relatively small, although the vocabulary of the recognition is virtually unlimited. In the system presented here, there are about 26000 morphs in the lexicon.

In decoder, the use of sub-word units implies that the word boundaries are not obtained automatically. The word boundaries are resolved by duplicating the paths with possible word boundaries, and inserting the word boundary to one of these paths. If there is acoustic silence between the morphs, the word boundary is always inserted.

From now on, to maintain the common terminology, the words “word” and “morph” are used interchangeably. Therefore, for example, the term “word history” should in this context be “morph history”, but the former is still used for clarity.

Another issue in Finnish speech recognition is that most of the Finnish phonemes have two length variants, distinguished from each other by different durations of the phones. To improve the discrimination between these phoneme variants, the HMM state durations are modeled explicitly using gamma distributions. In decoding, the duration probabilities were added to the total likelihood, similar to the post-processing approach in stack decoders (Pylkkönen and Kurimo 2004). As the acoustical differences between the phoneme variants are small, these phonemes are combined in triphone contexts so that the two variants are distinguished only in the central phoneme of the triphone.

3. Constructing the search network

In (Demuynck et al. 2000), a procedure for constructing a compact search network was presented. Here, a new and simpler scheme is presented, which still achieves a relatively compact representation of the constraints set by the lexicon and acoustic models. Instead of first building a context independent network, most of the search structure is constructed at once, and only minor post processing is required.

The search network is built around a popular idea of a lexical prefix tree. As suggested by Demuynck et al. (2000), the traditional phone-level tree can be made even more efficient by utilizing the HMM level state tying, which has been implemented here. Cross-word triphone contexts are handled by building a separate network, to which the lexical prefix tree is linked. The new network structure is very compact, and increases the size of the search tree only moderately.

We use the following terminology. The search network is built of *nodes*, which are linked to each other with *arcs*. Nodes can either correspond to one *HMM state*, or be *dummy nodes* without any acoustic probabilities associated with them. In decoding, these dummy nodes are passed immediately, they merely mediate the *tokens* used to represent

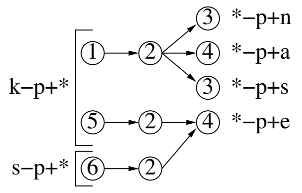

Figure 1: An example of the fan-in triphone node organization. The numbers inside the nodes represent the (shared) HMM state numbers. Triphone notation k-p+* represents /p/ after /k/ and followed by any phoneme.

the active search network. A node can also have a *word identity* associated with it, which leads to insertion of the word into the word history of the token passing that node.

Currently the construction procedure assumes triphone models, but it could be expanded to handle also wider contexts. Every triphone is defined in the acoustic models, and they are not tied at the triphone level. Instead, each triphone has a set of HMM states (currently three states in a left-to-right topology), and these states are shared among all triphones. The state tying has been performed using a decision tree.

Next, the steps in the construction of the search network are discussed in detail.

3.1. Creating fan-in triphones

The construction of the search network starts by creating the fan-in triphones. This means that the HMM state sequences of every triphone are inserted to the search network. They are organized by their central phoneme and the left context, so that if these are equal, the fan-in triphones are allowed to share their first nodes. The last nodes, however, are grouped differently, now according to the central phoneme and the right context. This way the number of arcs are minimized when linking other nodes to or from the fan-in triphones. An example of the resulting node connections is shown in Figure 1.

3.2. Construction of the lexical prefix tree

The lexical prefix tree is constructed by adding words to the tree one at a time. The words are first expanded to the corresponding HMM state sequence. The construction algorithm then starts from the (dummy) root node, and finds the path in the tree which is common to the given state sequence. When a node is reached from which the path can not be continued, the rest of the states are inserted as tree nodes starting from the last common node. For the first phonemes, the left context is silence, so in decoding the root node is accessed only after silence. Other contexts are handled by fan-in triphones, which are linked to the second phonemes of the words, as explained in the next subsection.

The states are added up to the second last phoneme of the word. From there an arc is made to a dummy node containing the word identity of the new word. This node is then linked to proper fan-out triphones, which are created on demand.

The fan-out triphones are organized the same way as the fan-in triphones so that triphones belonging to the same phoneme and having the same left context are grouped together and are allowed to share their common states. When the dummy node at the end of a word is to be linked to the fan-out triphones, the node is linked to every starting node of the corresponding group (defined uniquely by the last triphone of the word).

Single phoneme words must be handled separately. The only node linked to the root node is the dummy node with the word identity. This is linked to the fan-out triphones

Table 1: Search network statistics

	nodes	arcs
Fan-out	13162	60572
Fan-in	22471	84693
Prefix tree	198829	451295
Total	234462	596560

as explained above. However, for single phoneme words also another implementation of the word has to be added inside the cross-word network. This is done by adding a dummy node with the word identity after every fan-in triphone whose central phoneme corresponds to the given word. This dummy node is then linked back to the fan-in triphones, determined by the right context of the originating fan-in triphone.

3.3. Building the cross-word network

After the words have been added to the prefix tree and the fan-out triphones have been created, the last nodes of the fan-out triphones are linked to the corresponding fan-in triphones. This simply means adding arcs to the first nodes of the corresponding group of fan-in triphones. Remember that each fan-out triphone has their last nodes shared with the triphones having the same central phoneme and the right context. The corresponding group of fan-in triphones is therefore determined uniquely, and the number of arcs is minimized.

The fan-in triphones are then linked back to the lexical prefix tree to make the tree re-entrant. This is done similarly as linking the fan-out triphones to fan-in ones, but now the targets are the nodes belonging to the first states of the second phonemes of the words.

3.4. Post processing

In a post processing phase, the word identities are propagated towards the root of the node, to obtain unique word identity in decoding as early as possible. Also in post processing, the word identity tables for language model lookahead are collected.

Table 1 shows some statistics of a tree constructed using the 26k morph lexicon. There were 2165 different HMM states in 30000 triphones (48 central phonemes, 25 different left and right contexts). As can be seen in the table, the number of nodes in the cross-word network is quite small compared to the number of nodes in the prefix tree. These figures seem rather similar to the ones reported in (Demuynck et al. 2000), although it must be noted that the number of triphones and their tying can affect the structure of the search network greatly.

4. Language model lookahead

When the lexical network is constructed as a prefix tree, the word identities can be determined only after there are no more branches in the tree. Thus the inclusion of the language model (LM) probability is delayed. It is well known that using LM probabilities as early as possible enhances the beam pruning and therefore decreases the size of the search space. This can be achieved by applying a language model lookahead technique.

Ortmanns and Ney (2000) presented a method for computing LM lookahead scores using a compressed tree structure to propagate the scores of individual words to the nodes of the search network. In principle that method could compute the scores only for nodes for which the scores are needed. However, for our purposes, this kind of method seems to be overly complex, considering that to determine the language model scores at the beginning of the tree, all the nodes would have to be processed anyway. Therefore for our decoder, a simpler approach was adopted. After the lexical network has been built, a list of possible word identities which are reachable from each node are collected and saved to the nodes. The LM lookahead score can then be computed by finding the maximum of the LM scores over the words in the node's list.

To minimize the significant amount of redundant computations involved in the LM lookahead, we created a two level caching system. Each node contains a simple cache of the maximum LM scores of the possible follow-up words for different word histories. If the correct word history is not found in the cache, a higher level cache is referenced. It stores the LM scores of all the words for a certain number of previous word histories. In case of a cache miss, the probabilities of all the words in the LM for the given word history are computed and stored to the cache. With back-off language models, this computation can be done rather efficiently. During preliminary evaluation it was noted that for the LM lookahead bigram language model gave the best performance with respect to computational effort. For the final LM, a 4-gram model was used.

In (Ortmanns and Ney 2000) it was suggested that LM lookahead is used only in the first nodes of the lexical prefix tree. In a preliminary evaluation four node generations were found to be enough. The LM lookahead is also applied only in those nodes where the list of possible word identities has changed from that of the previous nodes. Reducing the number of nodes in which LM lookahead is applied helps to save memory when a node level caching is involved. With four generations, the final number of nodes with LM lookahead caches was 2503, when without this reduction 17087 node level caches would have been needed.

5. Experimental evaluation

The new decoder design was compared against our previous decoder (Hirsimäki and Kurimo 2004), which was based on the principle of stack decoding. Also the effect of the cross-word (CW) network was analyzed by running the new decoder with and without the cross-word triphone handling.

The evaluation task was a Finnish speaker dependent LVCSR task (Siivola et al. 2003). Preliminary testing and parameter optimization was done with a separate 19-minute development set of the same material. The length of the actual evaluation set was 27 minutes. The acoustic models were trained from a training set of about 12 hours.

The results are shown in Table 2. We use phoneme error rate as the main error measure, as it better indicates the recognition performance of a highly inflectional language than the traditionally used word error rate. The efficiency was measured by a real-time factor, which does not include the time used to compute the framewise state probabilities.

It can be seen that the use of cross-word triphones improves the recognition accuracy by about 26%. This is much more compared to the improvements reported in English (Sixtus and Ney 2002). The reason for this is probably the sub-word lexicon, which gives rise to a much higher amount of cross-word contexts. It is also remarkable that the use of cross-word triphones actually makes the decoder even faster. This contradicts the effect of using cross-word contexts with dynamically expanded search space (Sixtus and Ney 2002), showing the efficiency of our static expansion of the search network.

Table 2: Results of the decoder evaluation

Decoder	Phoneme error	Word error	Real-time factor
Stack decoder	2.58%	15.5%	4.3
New, no CW triphones	2.65%	15.5%	3.1
New, with CW triphones	1.96%	12.6%	1.3

6. Conclusions

This paper presented the design of a decoder which is able to handle cross-word triphone contexts efficiently using a statically expanded search network. The decoder also includes some language specific extensions necessary for Finnish LVCSR. The decoder uses a one-pass time-synchronous beam search, which is enhanced using a language model lookahead technique, also presented in the paper. The decoder was compared against our previous decoder, and the performance proves the efficiency of the new design.

7. Acknowledgments

This work was supported by the Academy of Finland in the project "New adaptive and learning methods in speech recognition".

References

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

Demuynck, Kris; Duchateau, Jacques; Compernolle, Dirk Van; Wambacq, Patrick 2000. An efficient search space representation for large vocabulary continuous speech recognition. In: *Speech Communication* 30, 37–53

Hirsimäki, Teemu; Kurimo, Mikko 2004. Decoder issues in unlimited Finnish speech recognition. In: *Proceedings of Norsig*. 320–323

Mohri, Mehryar; Pereira, Fernando; Riley, Michael 2002. Weighted finite-state transducers in speech recognition. In: *Computer Speech and Language* 16, 69–88

Ortmanns, Stefan; Ney, Hermann 2000. Look-ahead techniques for fast beam search. In: *Computer Speech and Language* 14, 15–32

Pylkkönen, Janne; Kurimo, Mikko 2004. Using phone durations in Finnish large vocabulary continuous speech recognition. In: *Proceedings of Norsig*. 324–327

Siivola, Vesa; Hirsimäki, Teemu; Creutz, Mathias; Kurimo, Mikko 2003. Unlimited vocabulary speech recognition based on morphs discovered in an unsupervised manner. In: *Proceedings of Eurospeech*. 2293–2296

Sixtus, Achim; Ney, Hermann 2002. From within-word model search to across-word model search in large vocabulary continuous speech recognition. In: *Computer Speech and Language* 16, 245–271

JANNE PYLKKÖNEN (M.Sc.) is a post-graduate student working as a researcher at the Neural Networks Research Centre of Helsinki University of Technology. E-mail: janne.pylkkonen@hut.fi.