A One Pass Semi-dynamic Network Decoder Based on Language Model Network*

Dong-Hoon Ahn and Minhwa Chung
Department of Computer Science, Sogang University
Sinsu-Dong, Mapo-Ku, Seoul 121-742, Korea
{drahn, mchung}@sogang.ac.kr

Abstract
Decoding in a precompiled static network, compared with one in a dynamically managed network, is easier to implement and faster enough to yield a near real time response. However, when the recognition system handles a complex task, it has a problem of intensive memory usage. To overcome this weakness, we present a new decoding strategy that combines the advantages of static and dynamic network architectures. In this strategy, we first define a language model (LM) network that can represent an arbitrary back-off N-gram in a finite state network (FSN). The LM network enables constructing a precompiled static network and partitioning the whole network into subnetworks using LM histories. Then the recognition network can be dynamically created and destroyed on the subnetwork’s basis. To make dynamic management of networks as simple as possible, we also devise a data structure for network representation that self-structures its nodes and arcs. The final decoder maintains subnetworks as needed, but does not need to maintain nodes and arcs. Experimental results show that this semi-dynamic management of networks dramatically reduces memory usage at the cost of less than 10% increase of recognition time.

1. Introduction
The FSN representation is frequently employed in the initial decoding steps of many speech recognition systems since the Viterbi-based decoding algorithm can be effectively performed on the FSN with great speed [1, 2, 3]. However, such a recognition network of the FSN style has inherent problems that its size drastically grows as the task gets more complex since it should integrate all the knowledge sources inside itself. The use of a lexical prefix tree [4] may give more compact representation to reduce the computational load and the size of the network itself, but another problem of the delayed application of LM has prohibited the use of the fully precompiled static network in many ways [2]. Many literatures have tried to tackle this problem [1, 2, 4, 5]. Especially in [1], they introduced an efficient way of representing a bigram and compiled into a static tree-based recognition network using successor trees and null nodes. The final network is fully static, reentrant depending on the linguistic context, and more importantly, of spatially manageable size. Naturally it has no overhead such as the dynamic instantiation [2] or tree-copying operations [4] during recognition, and even can be optimized in terms of FSN.

However it has two problems in the spatial aspect. The first is that successor trees significantly increase the memory requirement when the recognition network is built with higher order N-gram, and bigger vocabulary [5]. They and others [3] provided some optimization methods to reduce the size of network but the network minimized from an inherently huge one may still bear large spatial constraint. Unfortunately this phenomenon is inherent as the number of successor trees is proportional to the sizes of the LM and the vocabulary. The second problem is that although the static recognition network is precompiled with a large amount of memory, only a fraction of the whole network is used due to the tight beam pruning methods. From the viewpoint of token-passing paradigm [6], many tokens are dropped in the Viterbi competition and do not reach the end of successor trees. This means that there are many successor trees that have no tokens during recognition.

This paper proposes an efficient decoding architecture that can cope with these spatial drawbacks at the cost of a bit of increase in the recognition time. First we explicitly introduce a LM network, a generalized representation of [1], which can model arbitrary backoff N-gram with FSN. A LM network is easily precompiled into a fully static network and provides a reasonable way to partition the whole network into subnetworks with LM histories. Instead of fully instantiating the static network before recognition, we manage the network as needed during recognition. Again instead of creating and destroying all the nodes and the arcs individually, we do on the basis of the subnetworks. This is called a semi-dynamic network management. To make the management as simple as possible, we also devise a network representation that self-structures its nodes and arcs.

In the following, we introduce a LM network and the subnetwork-based representation of a recognition network. Then the method of token propagation with semi-dynamic management of networks is described. Finally experimental results are provided and discussed.

2. Language model network
Formally a language model network is a representation that models arbitrary N-gram so that the recognition network is transparently built from the specific language model. The LM probability or likelihood of word sequence $W=w_1w_2\cdots w_n$ is calculated as follows.

$$p(W) = p(w_1w_2\cdots w_n)$$
$$= p(w_1) \times p(w_2|w_1) \times \cdots \times p(w_n|w_1\cdots w_{n-1})$$
$$= \prod_i p(w_i|w_1\cdots w_{i-1}) = \prod_i p(w_i|h) \quad (1)$$

* This research was supported by Korea Science and Engineering Foundation(PN 2000-1-30300-003-3). Also, the authors would like to thank HCI Lab, SAIT for providing the speech DB used in the experiments.
h's are called LM histories, and usually divided into equivalence classes using equivalence relation. For example when we use N-gram model, histories are equivalent iff the last N-1 words are identical. Whatever equivalence relation or LM is used, there are two facts applied to all the language models. One is that a history is always present, and the other is that a pair of a history and each word following it becomes another history that will be followed by a set of words. A language model network can be defined from these facts. We define (h, w), a pair of history h and one word w following it by language model context. A node in the LM network corresponds to a language model context and an arc, to a transition between two contexts, weighted by its conditional log-probability. The log-likelihood of a word sequence is calculated from the accumulation of the weights on the arcs on paths followed. There are two distinguished nodes: an entry and an exit node in a sentence. Particularly all the nodes include an arc to the exit node since any word can be the final one in a sentence. A backoff arc is added to account for the unseen word sequences. Its destination node has the backoff history of the source node and is weighted by a backoff weight. As [1] employed a null node to represent the backoff transition, a backoff arc is treated as null transition. When a node has null transition, a process passes by the node and reaches the next destination. Once a network is built, a network reduction algorithm [1] is applied to it.

Figure 1 is an example network representing trigram model with two words {a, b}. For simplicity an exit node and arcs incoming to it are omitted. (0, 0) is an entry node with “no” history and “no” last input word. A node labeled with the unique number is actually a set of LM contexts that proved to be equivalent by the reduction algorithm.

![Figure 1. A language model network with backoff.](image)

( p(·) : a LM probability, b(·) : a backoff weight)

3. Subnetwork-based representation

A set of words that follow a specific LM history or context can be identified from the arcs that follow the corresponding node in a LM network. These words are called successor words in [1] and precompiled into a successor tree. A fully static recognition network can be easily constructed with these successor trees.

This network is now partitioned into a number of subnetworks. A subnetwork is defined as a partitioning unit that simplifies a dynamic network management. The recognition network may be partitioned in several ways, but there should be a manageable number of subnetworks. If there are too many, the management cost might slow down the decoding. However if there are too few then the amount of spatial reduction from the dynamic management could be negligible. Finally a subnetwork should be less related with others than inside; if they were closely related, they would activate each other frequently and then this could increase the management cost. Considering these conditions, a successor tree is well fitted into our purpose. The number of successor trees is far less than that of HMM instances and it is associated with a LM history that defines a set of successor words and provides a convenient way to estimate how frequently it will be activated during recognition, which will be exploited later.

When the network is dynamically managed, it should be carefully done so that the time taken to decode is not noticeably increased. That means both the cost and the frequency of constructing and destroying a subnetwork on the fly should be minimally maintained. In this section we investigate the way to minimize the cost first. The next section will provide a solution to the frequency problem.

Generally a network is defined with a set of nodes and a set of weighted or unweighted arcs and implemented with separate structures. Considering the dynamic network management of individual structures, there may be two major problems. One is the memory fragmentation problem induced by a number of small objects such as nodes and arcs. Obviously the memory manager dedicated to the system or supported by OS can require more computations during recognition. The second problem is that there are too many nodes and arcs to manage. When a subnetwork is constructed or destroyed its nodes and arcs should be done accordingly. It is certainly nontrivial.

We resolve these problems by making a subnetwork self-structuring. A self-structuring network is one whose elements are automatically structured or instantiated instead of being individually constructed when the network gets instantiated. Figure 2 (a) shows an example subnetwork after LM probability have been factored in it [1], and figure 2 (b) shows an internal structure of a subnetwork. The nodes, the arcs and the non-zero weights are separately stored in the individual sets and indexed by unique numbers, not by physical addresses. Each set can be found with the offsets located in the initial part of the network. By putting all into the sets, we can resolve the fragmentation problem.

A node includes two pairs of indexes. One is (the beginning index of an arc outgoing from the node, the number of outgoing arcs), and the other is (the beginning index of a weighted arc outgoing from the node, the number of weighted outgoing arcs). As for an arc, it has an index for the target subnetwork or node according to whether it is from the word-end node or not. Furthermore an arc has a variable length; non-zero weights are stored in the weight set, but zero weights are not. Since the number of weighted arcs is typically small, such a representation gives more spatial savings.
Due to the index-based representation, the address of an element can be easily obtained with its offset plus the starting address of the subnetwork and this provides quite a simple mechanism to instantiate a subnetwork. When a subnetwork is complete, a chunk of memory occupied by it is written on a physical disk as is. Then the procedure to instantiate a subnetwork is taken as follows.

- Allocate memory as much as its size at one time.
- Read into the memory from where the chunk has been written.
- Calculate the real addresses of the sets using the physical address of the subnetwork and offsets to them.
- Put them back where offsets are located so that all the nodes and arcs are identified with its index in the set.

The term semi-dynamic management comes from the fact that the subnetwork is dynamically managed, but the nodes and the arcs do not need creating and destroying on the fly.

4. Semi-dynamic network decoder

4.1. Semi-dynamic network management

The basic search method is a one-pass Viterbi decoding based on the token-passing paradigm. When the static recognition network is ready, all that a decoder does is to propagate tokens into the network. However when the network is to be dynamically managed the decoder should know the state of the network. As a result, the decoder should activate subnetworks to be used at hand, but have not been instantiated yet. As pointed out in the previous section, the frequency of activating and deactivating subnetworks should be as low as possible.

To relax this problem we adopt a caching strategy so that frequently activated subnetworks are in memory, but others are not. Besides we preload some of the subnetworks to reduce the decoding time. The followings are preloaded.

- Two subnetworks that include the sentence entry and exit nodes.
- A subnetwork that follows the sentence entry node.
- A unigram subnetwork.
- Subnetworks likely to be frequently activated.

The first three types of subnetworks are selected as the minimal set. The problem is to select the fourth type of subnetworks. In the following our approach is described.

4.2. Estimation of activation frequency

We estimated the activation frequency of a subnetwork by the likelihood of a LM history associated with it. When its estimate is greater than a pre-specified $p$-threshold the subnetwork is preloaded. For example, consider two subnetworks. One has the words that follow the sequence in the and the other, the sequence my diary. Normally the former sequence has more likelihood than the latter. Also we may expect the former would be more frequently shown and more frequently activate the corresponding subnetwork.

Calculating the likelihood of a LM history $h$ is simple. When $h = ab$, $p(h) = p(ab) = p(a)p(b|h)$ is calculated. Fortunately the LM network already has the terms needed to calculate on each arc. While the LM network is built we add the weight on an arc to the accumulated weight in its starting node and store it in the destination node. If two or more arcs have reached the same node the best is kept.

4.3. Token passing with dynamic network management

The token passing algorithm [6] is slightly modified so that a dynamic management is employed. The first token is initiated from the entry node. When tokens are passed to the next subnetwork at the word-end, the decoder should check if the subnetwork is already instantiated. In case of its availability tokens are propagated. Otherwise the decoder loads the subnetwork to instantiate from the disk and propagates tokens through the new subnetwork. To prevent a monotonic increase in the number of subnetworks, a subnetwork should be withdrawn from the memory when it has no tokens inside and is less likely to get active ones again in the near future. For this purpose we use aging strategy; if there is no token during a pre-defined interval of $k$-threshold (in frame) since the last active one is dropped, we can safely destroy the subnetwork. In fact many words except function words tend to appear at most once or twice in the sentence, thus once a word has fallen it rarely seems to revive.
5. Experimental results

5.1. Test conditions

Our decoder used acoustic and language models trained using the database of read speech in Korean. It amounts to 25 hours and comprises 22k words. We used 48 phoneme-like-units and morpheme-based decoding units. An acoustic model is a set of tied-state CDPHMM with each state of 12 Gaussian mixtures. The perplexities for 154 sentences selected for testing are 151 for bigram and 131 for trigram.

5.2. Comparison of network structures

Table 1 shows the structure of baseline and equivalent subnetwork-based recognition network for trigram. We can learn the subnetwork-based representation is more compact, especially due to the variable length representation of an arc.

Table 1: The comparison of network structures

<table>
<thead>
<tr>
<th>node</th>
<th>count</th>
<th>size (MB/Mega Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>arc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>1455979</td>
<td>5071048</td>
</tr>
<tr>
<td>subnetwork</td>
<td>77.5MB</td>
<td>47.7MB</td>
</tr>
</tbody>
</table>

5.3. Dynamic management cost for each configuration

Both the baseline and the subnetwork-based recognition network have the same word accuracy of 81.61% with trigram. In the table 2, the semi-dynamic management cost is investigated with respect to the averages of RTF (real time factor), memory size and H/R (hit ratio on the subnetworks in memory) when several combinations of p- and k-thresholds are given. Experiments are conducted right after the system is reset so that they are not affected by the cache contents of physical disks.

Table 2: The cost of semi-dynamic management

<table>
<thead>
<tr>
<th>k</th>
<th>RTF</th>
<th>p=8</th>
<th>p=10</th>
<th>p=12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.032</td>
<td>2.028</td>
<td>2.041</td>
<td>2.032</td>
</tr>
<tr>
<td>MB</td>
<td>4.944</td>
<td>5.237</td>
<td>7.890</td>
<td>18.093</td>
</tr>
<tr>
<td>H/R</td>
<td>0.932</td>
<td>0.934</td>
<td>0.939</td>
<td>0.951</td>
</tr>
<tr>
<td>10</td>
<td>2.009</td>
<td>2.005</td>
<td>1.973</td>
<td>1.971</td>
</tr>
<tr>
<td>MB</td>
<td>5.493</td>
<td>5.818</td>
<td>8.767</td>
<td>20.104</td>
</tr>
<tr>
<td>H/R</td>
<td>0.939</td>
<td>0.941</td>
<td>0.946</td>
<td>0.958</td>
</tr>
<tr>
<td>100</td>
<td>1.941</td>
<td>1.932</td>
<td>1.941</td>
<td>1.897</td>
</tr>
<tr>
<td>H/R</td>
<td>0.953</td>
<td>0.955</td>
<td>0.960</td>
<td>0.972</td>
</tr>
<tr>
<td>baseline</td>
<td>1.864</td>
<td>MB</td>
<td>77.5</td>
<td></td>
</tr>
<tr>
<td>all loaded</td>
<td>RTF</td>
<td>1.843</td>
<td>MB</td>
<td>47.7</td>
</tr>
</tbody>
</table>

\(^1\) When the minimal set of subnetworks are preloaded.

\(^2\) When all the subnetworks are preloaded.

As shown in the table, while the real time factor has increased at most by 10%, the amount of memory is significantly decreased; the least case needs only 6% relative to the size of baseline and 10% relative to that of fully loaded subnetworks. A hit ratio is gradually increasing as the p-threshold falls and it proves the usefulness of the likelihood of a LM history used in the estimation of activation frequency.

Figure 3. RTF ratio for the first 100 sentences.

The last graph shows the overall behavior of the dynamic management procedure for the first 100 sentences. The RTF ratios are calculated with dividing the RTFs of sentences in the test set by those in the ‘all loaded’ set. The minimal set of preloaded subnetworks has a very high peak at the initial phase, but the ratio converges on some point. Others are similar but the peak goes down as more subnetworks are preloaded. This means that there are a high rate of activities to instantiate and remove a large number of subnetworks during initial phase, and by doing so the dynamic management fits the recognition network into the task session.

6. Conclusions

We presented a semi-dynamic network decoder within the token-passing framework. The novel LM network is introduced. An efficient subnetwork-based representation and the semi-dynamic network management method are established based on it. Experimental results show that the proposed network management method reduces the memory requirement significantly with a bit of computational cost. Further refinements are needed so that a subnetwork can handle crossword triphone network.

7. References