ICASSP 97, Munich, pp. 1787~1790, 1997

1

AN EFFICIENT SEARCH METHOD FOR LARGE-VOCABULARY CONTINUOUS-SPEECH RECOGNITION

Ken Hanazawa, Yasuhiro Minami**, and Sadaoki Furui* ***

*Tokyo Institute of Technology
Meguro-ku, Tokyo, 152 Japan
**NTT Human Interface Laboratories
Musashino-shi, Tokyo, 180 Japan

ABSTRACT

This paper proposes an efficient method for large-vocabulary continuous-speech recognition, using a compact data structure and an efficient search algorithm. We introduce a very compact data structure DAWG as a lexicon to reduce the search space. We also propose a search algorithm to obtain the N-best hypotheses using the DAWG structure. This search algorithm is composed of two phases: "forward search" and "traceback". Forward search, which basically uses the time-synchronous Viterbi algorithm, merges candidates and stores the information about them in DAWG structures to create phoneme graphs. Traceback traces the phoneme graphs to obtain the N-best hypotheses. An evaluation of this method's performance using a speech-recognition-based telephone-directory-assistance system having a 4000-word vocabulary confirmed that our strategy improves speech recognition in terms of time and recognition rate.

1. INTRODUCTION

Several efficient search algorithms for large-vocabulary speech-recognition systems have been proposed: A* or stack decoding [1, 2], using word graphs [3] or word lattices [4], and recently, combining some search algorithms [5]. However they always use a tree structure for the word lexicon, so when speech recognition treats a lot of words that have the same suffix (for example the name of a company and a person's name), a tree structure is not so efficient because it does not merge suffix parts. We introduce a network structure as the word lexicon, which is more compact than a tree structure.

"DAWG" (directed acyclic word-graph) data structures [6] have been used in the natural-language field for checking spelling and retrieving data from a database efficiently. Several methods have been proposed for generating DAWG structures. However, they can only be used to generate networks all at once. This means that when we want to add or delete a word, we have to regenerate the DAWG structure. For these methods to be used for speech recognition, they should be able to add new words or delete unnecessary words as fast as possible. The method proposed by Aoe [7] to solve this problem can add or delete a word quickly. We introduce this method to generate DAWG structure.

2. TELEPHONE-DIRECTORY-ASSISTANCE TASK

To easily address our method, we explain about our task first. Our task is a telephone directory assistance task, but we note that our method is applicable to more than this task; it can be used for general data retrieval tasks.

Our system recognizes five classes of keywords. These classes are necessary to retrieve data related to telephone numbers: Prefecture, City, Town, Block number, and Subscriber Name. A sentence to be recognized is a combination of these keywords and non-keywords between items. Non-keywords are for accepting utterances that have interjections and requirement expressions, for example, "Sumimasen etto, Tokyo no Mitaka-shi, etto Minami-san no denwabangou wo oshietekudasai" (In English: "Excuse me, uh could you please give me the phone number of uh Mr. Minami in Mitaka, Tokyo?").

3. NETWORK STRUCTURE

Our telephone-directory-assistance task has five keyword classes for names and addresses. Our grammar consists of two parts: a main grammar and several sub-grammars (Fig. 1). The main grammar controls the relationships between the sub-grammars according to the "meaning", where meaning is defined as a combination of keywords, for example, {city, name}, {town, block number, name}, and {town, name}. The main grammar is represented by a kind of tree structure. The sub-grammars handle the keyword classes and the non-keyword class. Networks of these sub-grammars are constructed using DAWG structures.

A DAWG structure is more compact than a tree structure. The "Trie" structure [8], which is a kind of tree structures, is used for data retrieval in the natural language processing field. This Trie structure merges the common prefixes of the word lexicon. A DAWG structure is obtained by merging the common suffixes of the Trie structure. The DAWG structure can thus generate a very compact lexicon. Figure 2 shows an example of the Trie structure for keyword *K* and the DAWG structure for the same word. The networks are composed of nodes and arcs. Phonemes are assigned to the arcs. We can reduce the search space and memory space significantly by using a DAWG structure. However, it is difficult to manually generate this structure for huge amounts of data. We therefore need a method of generating it automatically. Several methods have been proposed for generating DAWG structures, but we use the method proposed by Aoe to generate the lexicon network because it can generate the DAWG structure automatically.

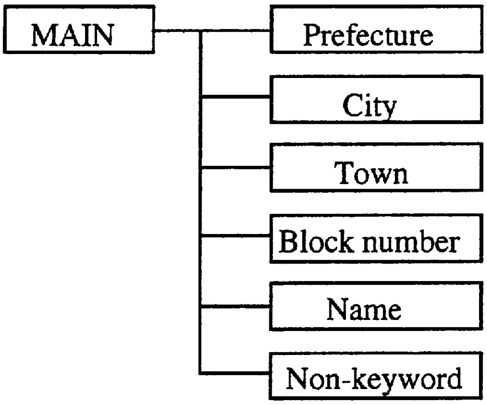
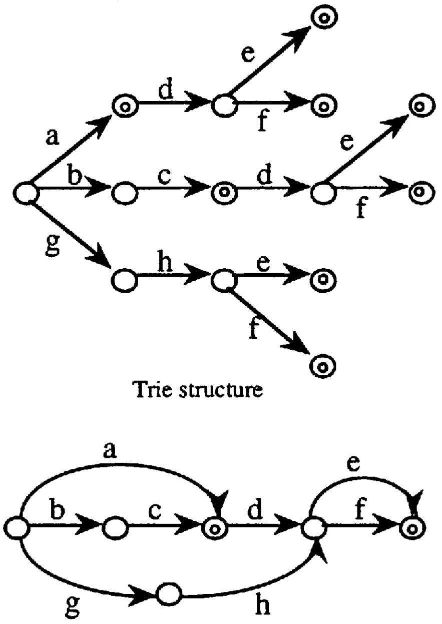


Figure 1. Main grammar and sub-grammars.



Trie structure

DAWG structure

K={a, ade, adf, bc, bcde, bcdf, ghe, ghf}

◎ terminal node

Figure 2. Examples of Trie and DAWG structures for keyword K.

4. SPEECH RECOGNITION SYSTEM

When applying DAWG data structures to a speech recognition system to produce the N-best hypotheses, a new search strategy is required because networks are merged at many nodes, for example, the nodes after arcs "c", "d", and "f" in Fig. 2. Also, to produce the N-best hypotheses, we have to trace the networks backwards.

Our search method has two phases: forward search and traceback. The basic concept of forward search is similar to that of making a word lattice [4]. However, our forward search creates phoneme graphs instead of word lattices. Traceback searches the phoneme graphs to produce the N-best hypotheses.

Forward search

Our recognition system searches the DAWG networks by using the time synchronous Viterbi search algorithm. Its algorithm is as follows.

```
0 Initialize. Put a candidate into the beginning node of the
  main network.
1 For the main network
    1.1 For all the candidates on nodes of the main network,
        pass them to the beginning nodes of all possible
        keyword and non-keyword networks.
    1.2 If multiple candidates arrive at the same beginning
        node of a keyword or non-keyword network, they
        are merged.
2 For keyword and non-keyword networks
    2.1 For all the candidates on the nodes of keyword and
        non-keyword networks:
          2.1.1 Time synchronous Viterbi decoding is
                performed for all the candidates.
          2.1.2 If a candidate does not have a high enough
                score, it is pruned.
    2.2 For all the candidates on the nodes of keyword and
        non-keyword networks:
          2.2.1 If multiple candidates arrive at the same
                node, they are merged.
          2.2.2 If a node is one of the terminal nodes of its
                network, pass it to the corresponding node
                of the main network. Here, merging is also
                performed.
3 If it is not the end-time, go back to 1. Otherwise, stop.
```

Merge: When multiple candidates arrive at the same node at the same time and only if their meanings are the same, only the candidate with the best score is retained and the information that it arrived at the node is stored. For the other candidates, information about them is just stored.

Merging multiple paths into one path reduces search costs. Since a candidate does not have a history of its phoneme sequence, but only has a score of the candidate, the memory requirements are also reduced during Viterbi decoding. A sentence is constructed using traceback. The information stored at each node is the time when the path arrived at the node, the score of the path, the previous node, and the meaning of the path. Candidates are sorted in descending order of their scores.

A merging is also performed in the main grammar: multiple paths arriving at the same node at the same time are merged. These paths are merged if they have the same keyword classes, even if the keywords are different. The information about the paths is stored at that point.

Traceback

After the forward search, only the maximum score is obtained. So, to obtain the N-best hypotheses, the phoneme graphs should be traced, all the phoneme sequences should be constructed, and their scores should be compared.

All the information we need for traceback on a node is when, from where, and with which meaning a path came to the node. Since all of these items were already stored and sorted on the nodes of phoneme graphs, the main algorithm for traceback is simple. The traceback process traces these graphs in the opposite direction (backwards) and generates all the candidates. Then it outputs the top N candidates of these as N-best hypotheses. However, it is not efficient to create all candidates exhaustively. To avoid this problem, we introduce the following method. First, a buffer of N hypotheses is prepared. The phoneme graphs are traced from the end of the sentence, candidates are constructed at every node, looking at the stored information in descending order. The generated candidates are added to the buffer so that the scores of candidates come in a descending order. If the score of a candidate is smaller than that of the Nth candidate in the buffer, the candidate is discarded and successive nodes will not be checked any more, because scores of the candidates generated from this candidate in the future will not exceed the score. This operation is done recursively for all nodes.

When multiple candidates that have the same keywords are generated by a small difference in the paths, only the candidate that has the maximum score is added to the buffer.

Traceback can evaluate all meanings and all information stored during the forward search.

5. EXPERIMENT

The vocabulary contains more than 4000 words in the keyword classes. Table 1 shows the number of keywords contained in each keyword class.

Table 1. Number of keywords.

Keywords	Numbers
Prefecture	2
City	6
Town	40
Block Number	1,223
Name	3,505

In the main grammar, we only permit occurrences of each keyword either once or not at all in a sentence. A name has to appear once in a sentence, because without names, we cannot get telephone numbers. We consider a sentence which does not keep the rules of the main grammar as Out-Of-Syntax.

We considered the recognition results to be "correct" if all the keywords in the sentence were recognized correctly.

5.1. Training and Test Data

The phoneme HMMs that are context dependent have four states, three loops, and four Gaussian mixtures. The feature parameters are 16 *cepstra*, 16 Δ*cepstra*, and a Δ*power*.

The phoneme HMMs were trained by embedded training using 9600 phonetically balanced sentences uttered by 34 male and 30 female speakers.

We have two sets of test data. One (set A) has 362 sentences, and the other (set B) has 1045 sentences including the previous 362 sentences set. The sentences of set A were uttered by 12 male and 8 female speakers, and those of set B were uttered by 33 male and 13 female speakers. Table 2 shows them. All the speech utterances were collected by the multi-modal telephone-directory-assistance system [9]. Each speaker made about twenty attempts to get telephone numbers using information indicated on simplified city maps. Subscriber name and address were indicated on each map.

Table 2. Test data sets.

	sentences	speakers
set A	362	20
set B	1,045	46

In set A, the utterances sometimes had several non-keywords that should be accepted by the system: interjections, verb phrases, particles, etc. However, these utterances did not contain unknown words, repeats, or restarts. In set B, the utterances did include several non-keywords, and sometimes, words that were unknown. And there were some utterances contain unknown words (Out-Of-Vocabulary), Out-Of-Syntax, repeats, restarts, or other irregular expressions.

5.2. Recognition Results

Our recognition experiments were performed under the above conditions. After comparing the numbers of nodes in the Trie and DAWG structures, we show the experimental results for test set A compared with our previous system [10], and then we show the results for test set B and further experiments.

5.2.1. Number of Nodes

Table 3 shows the number of nodes for the Trie and DAWG structures. Each keyword class is represented by a single network. On a time synchronous recognizer, the number of nodes corresponds to the size of the recognizer's search space. As Table 3 shows, the search space is reduced significantly by using the DAWG structure.

Table 3. Number of nodes.

Network	Trie structure	DAWG structure
Prefecture	18	7
City	38	17
Town	665	115
Block number	36,987	602
Name	82,848	7,824
Non-keyword	968,639	169

To create the DAWG in the figure, we made possible phoneme sequences including allophonic differences from the CFG that we used in our previous system. That CFG was made by hand, looking at the simulated dialogs between operators and customers. During this creation, recursion was restricted, so we were able to make a finite number of sentences from the CFG.

We were surprised that the number of tree structure nodes exploded in non-keywords, while, the DAWG structure on the other hand, had only 169 nodes. This reason may be as follows: the Japanese language allows various expressions at the end of a sentence, so a tree structure has many branches there. Although this can be avoided by dividing the tree structure into several sub structures, doing this requires extra memory space during Viterbi process to store the history of the paths.

It was impossible to try to compare speech recognition using the Trie and DAWG structures because of the memory size of the Trie structure. However in time synchronous speech recognition, the node size almost represents search space size.

5.2.2. Comparison with Previous System

In the previous section, we said that we could not run our system using the Trie structure because of the explosion in the number of nodes. So we could not construct a time synchronous recognizer using the Trie structure (or tree structure). But the DAWG structure not only facilitated the time synchronous recognition, but because of the reduction in the number of nodes, it is expected to reduce the recognition time drastically. Instead of comparing the systems using Trie and DAWG structures, we compare our new method with our previous method, which uses the phoneme synchronous trellis-search algorithm. In this algorithm, a generalized LR parser is used as a language model for prediction; this parser can analyze context-free grammar. It merges multiple candidates if they have the same grammatical meaning.

Using test set A, the sentence-correct rates were 86.2% for the top choice and 96.7% for the top five choices. The word-correct rate was 94.4% for the top choice. Here, we only consider keywords for the word-correct rate. The average run time was about 56 seconds per sentence (HP9000J210). The run time for merging and sorting the candidates was about 1% of the CPU time.

With our previous system, its sentence-correct rates were 83.2% for the top choice and 91.5% for the top five choices. The word-correct rate was 92.8% for the top choice. The average run time was about 334 seconds per sentence. Thus, the new system has a higher sentence-correct rate than with the previous system, and it is about six times faster than the previous system using the same CPU.

5.2.3. Extended Experiments

We want our system to be robust for more natural speech. Therefore, we performed some extra experiments on test set B. Since set B includes some utterances which contain unknown words, repeats, restarts, etc., recognition performances for set B will be worse. We have to solve these problems of natural speech.

First, the main grammar was extended to reduce Out-Of-Syntax errors. We permit any sequences of keywords in a sentence even if they do not have a name. The number of Out-Of-Syntax sentences is reduced from 32 to 1 after this extension.

Second, the networks of keywords were extended to reduce Out-Of-Vocabulary errors. In the Japanese language, Kanji characters have several different pronunciations. Many errors result from this problem, so we added some supplementary ways to pronounce the keywords. And we also added some non-keywords that occurred in set B. Using Aoe's DAWG method, the cost of extending networks was very small when we added new words or new ways to pronounce keywords. The number of sentences including Out-Of-Vocabulary was reduced from 50 to 14 after this extension.

Table 4 shows results of sentence recognition experiments using test set B. They are: without extension, with extended main grammar, and with extended main grammar and networks.

Table 4. Recognition performances on set B.

	sentence-corr.		word-corr.
extension	top choice	top5 choices	
none	74.2	84.6	89.8
grammar	75.7	86.2	89.4
grammar/ network	77.3	89.2	90.1

As Table 4 shows, the recognition performance improved when we extended the grammar and networks. The costs for extending the grammar and networks were very low. Even when the grammar and networks were extended, the average run times per sentence were almost the same (Table 5).

Table 5. Average run time per sentence.

extension	av. time [sec]
none	56
grammar	61
grammar/network	61

Note that although we extended the grammar and networks, and the number of errors resulting from Out-Of-Vocabulary and Out-Of-Syntax was reduced, there were still some. This will be the subject of our next work.

6. CONCLUSION

We have introduced an efficient data structure for telephone-directory-assistance tasks. Our proposed search algorithm for this structure outputs the N-best results. Experimental results confirmed that our new system is superior to our previous system in terms of processing time and recognition rate.

We also examined our new system on more natural speech data. To enhance the performance, we extended the main grammar and networks. We could get better performance with little extra cost.

This method is expected to be applicable to general tasks in large-vocabulary speech-recognition systems.

REFERENCES

[1] P. Kenny, et al, "A*- admissible heuristics for rapid lexical access", *Proc. ICASSP'91*, May 1991, pp. 689-692.

[2] D. B. Paul, "Algorithms for an optimal A* search and linearizing the search in the stack decoder", *Proc. ICASSP'91*, May 1991, pp. 693-696.

[3] X. Aubert, and H. Ney, "Large vocabulary continuous speech recognition using word graphs", *Proc. ICASSP'95*, Vol. 1, May 1995, pp. 49-52.

[4] C. H. Lee, et al, "Automatic speech and speaker recognition", pp.429-456.

[5] P. S. Gopalakrisnan, L. R. Bahl, and R. L. Mercer, "A tree search strategy for large vocabulary continuous speech recognition", *Proc. ICASSP'95*, Vol. 1, May 1995, pp. 572-575.

[6] E. Fredkin, "Trie memory", *Commun. ACM*, 3, 9, Sept. 1960, pp. 490-550.

[7] J. Aoe, K. Morimoto, and M. Hase, "An algorithm of compressing common suffixes for trie structures", *Trans. IEICE* Vol. J75-D-II No. 4, April 1992, pp. 770-799.

[8] A. W. Appel, and G. J. Jacobson, "The world's fastest scrabble program", *Commun. ACM*, 31, 5, May 1988, pp. 572-578.

[9] O. Yoshioka, Y. Minami, and K. Shikano, "A multi-modal dialogue system for telephone directory assistance", *Proc. ICSLP'94*, Sept. 1994, pp. 887-890.

[10] Y. Minami, et al, "Large-vocabulary continuous speech recognition algorithm applied to a multi-modal telephone directory assistance system", *Trans. Speech Communication* 15, 1995, pp. 301-310.