Large Vocabulary Word Recognition based on a Graph-Structured Dictionary

Katsuki Minamino

D21 Laboratory, SONY Corporation
6-7-35, Kitashinagawa, Shinagawa-ku, Tokyo, 141 Japan

ABSTRACT

In this paper, a structural search using a word-node graph is proposed to speed up the isolated word recognition based on hidden markov models (HMMs). We define a distance measure for comparing pairs of words, and construct a graph keeping the structure of word distribution. Based on this graph, the number of words to be examined are restricted. Experiments show that the search complexity is considerably reduced with little degradation of the recognition accuracy.

1. Introduction

In isolated word recognition [1], the goal is to find the word $w_i$ maximizing the probability $P(w_i|y)$ for an observed sequence $y$. This is, in general, performed by using the phoneme based HMMs, a lexical tree, and the Viterbi algorithm [2]. The computational load has two major parts: local likelihood calculation, and trellis search. If a continuous distribution is used for the output probability, it is important to reduce the first part, that is, the likelihood calculation for all the probability density functions. For this problem, an approach using a tree-structured probability density function [3], or vector quantization [4] was proposed.

As for the second part, if full search is used, the complexity increases linearly with the vocabulary size. It is, therefore, necessary to reduce the search complexity for large vocabulary speech recognition in a real-time system. Beam search is a well-known technique. However it has the problem that search errors are caused by pruning in excess. Another approach is A"* search (e.g., a tree-trellis search [5]). The efficiency of this technique is dependent on the heuristics. The other approach is that we preliminarily select words to search, and replace the problem by small vocabulary word recognition. In this approach, it is necessary not to leave out the words $w_i$ with high probability $P(w_i|y)$.

In this paper, we propose new search algorithm using a word-node graph. This is a word pre-selection technique based on the word distance measure. To put it shortly, given a sequence $y$, we first calculate the probability $P(w_j|y)$ for some words $w_j$; secondly, we calculate the probability $P(y_i|w_j)$ of the words $w_j$ that are similar to the words $w_j$ with high probability. For this purpose, we define a distance measure between comparing pairs of words, and introduce a space for words based on the distance. Next, we construct a directed graph representing word distribution, where the nodes correspond to words. In the recognition process, the graph is used for the word selection, so that the search space is restricted.

In section 2, we define a new distance measure between word pairs. In section 3, we describe the method of constructing the word-node graph. In section 4, we present the recognition procedure using this graph. In section 5, experimental results are shown. Finally, we summarize our conclusion.

2. Word Distance Measure

2.1. Definition

In this section, the distance measure between word pairs is defined for the sake of representing word distribution. We consider a word set $\{w_1, w_2, \ldots, w_N\}$ of the vocabulary size $N$. Let $y_k$ be the observation symbol sequence set corresponding to the word $w_j$. The parameters of HMMs are generally trained to maximize the log likelihood $\sum_{k=1}^{n} \log P(y_k|w_j)$ for the data set $\{y_k \in Y_k \ | \ k = 1, \ldots, n\}$. As a result, we obtain the word model $\hat{w}_j$ corresponding to each word $w_j$. Then, the recognition process is carried out by calculation of model likelihoods $P(y|\hat{w}_j)$, $1 \leq j \leq N$.

Now we consider the expectation of $\log P(y|\hat{w}_j)$ under the fixed model $\hat{w}_j$, i.e.,

$$E[\log P(y|\hat{w}_j)] = \int P(y|\hat{w}_j) \log P(y|\hat{w}_j) dy.$$  \hspace{1cm} (1)

where $E[ \ ]$ is the expectation operator, and $y_k$ is the observation symbol sequence corresponding to the word $w_j$. This is approximately computed by using the observed data set $\{y_k \in Y_k \ | \ k = 1, \ldots, Q\}$ as follows:

$$E[\log P(y|\hat{w}_j)] \approx \frac{1}{Q} \sum_{k=1}^{Q} \log P(y_k|\hat{w}_j).$$  \hspace{1cm} (2)
If the model parameters are well estimated, then
\[
E[\log P(y_t | \hat{w}_i)] \geq E[\log P(y_t | \hat{w}_j)]
\]  
with equality if and only if \( \hat{w}_i = \hat{w}_j \). Furthermore, it seems reasonable to suppose that if \( E[\log P(y_t | \hat{w}_i)] \) is large, then \( \log P(y_t | \hat{w}_j) \) may be also large for \( y \in Y_t \).

Therefore, we define the distance measure \( D \) from \( w_i \) to \( w_j \) by
\[
D(w_i, w_j) := E[\log P(y_t | \hat{w}_i)] - E[\log P(y_t | \hat{w}_j)].
\]  
The distance measure \( D \) has two properties:
\[
\begin{align*}
(i) & \quad D(w_i, w_j) \geq 0. \\
(ii) & \quad D(w_i, w_j) = 0 \iff w_i = w_j.
\end{align*}
\]  
We should notice that this measure is directed, relative, and non-symmetric. If the distance \( D(w_i, w_j) \) is small, then \( E[\log P(y_t | \hat{w}_i)] \) is as large as \( E[\log P(y_t | \hat{w}_j)] \). This implies that the word \( w_j \) may have high probability \( P(w_j | y) \) for \( y \in Y_t \).

Next, we shall focus on the calculation of the distance \( D \). The calculation of \( E[\log P(y_t | \hat{w}_i)] \) using the equation (2) needs the observed sequence set \( \{ y^q_t \} \) \( q = 1, \ldots, Q \). However, in large vocabulary word recognition using phoneme-based HMMs, such a sequence set \( \{ y^q_t \} \) is not always available in practice. We wish, therefore, to calculate the expected value \( E[\log P(y_t | \hat{w}_i)] \) by using only the model parameters.

### 2.2. Approximation

We assume that there exists a typical symbol sequence \( E[y_t] \), which corresponds to the word \( w_i \), satisfying the approximation
\[
E[\log P(y_t | \hat{w}_i)] \approx \log P(E[y_t] | \hat{w}_j).
\]  
Our concern is to generate such a sequence \( E[y_t] \) based on HMM parameters.

Assume that each HMM is a left-to-right model with no skipped state. Let \( a_{ij} \) be a state transition probability, and \( b_{ij}(k) \) an observation symbol probability. Based on \( a_{ij} \), the probability of duration \( n \) in state \( i \) is given by
\[
(a_{ij})^{n-1}(1 - a_{ij}).
\]  
Thus the expectation of \( n \) is
\[
E[n] = \sum_{n=1}^{\infty} n(a_{ij})^{n-1}(1 - a_{ij}) = \frac{1}{1 - a_{ij}}.
\]  
Based on the expected value \( E[n] \), the state transition sequence \( X_{t} = x_{1t}, x_{2t}, \ldots, x_{T} \) is uniquely decided for the word \( w_i \), where \( T \) is the length of the sequence \( X_t \). If the symbol sequence \( Z_{t} = z_{1t}, z_{2t}, \ldots, z_{T} \) is generated according to the state transition sequence \( X_{t} \), then the probability \( P(Z_{t} | \hat{w}_i, X_{t}) \) is given by
\[
P(Z_{t} | \hat{w}_i, X_{t}) = a_{x_{1t}, x_{2t}} \cdot a_{x_{2t}, x_{3t}} \cdots a_{x_{T-1t}, x_{Tt}}.
\]  
Now we generate the symbol sequence \( Z_{t} \) such that the probability \( P(Z_{t} | \hat{w}_i, X_{t}) \) is maximized. It is easily decided as the sequence of symbol \( z_{m} \) maximizing each probability \( b_{x_{m}, z_{m}}(z_{m}) \). If the observation symbol probability distribution is a Gaussian distribution, then the symbol \( z_{m} \) is uniquely decided as an expected symbol. Moreover, the sequence \( X_{t} \) is an expected sequence based on the state transition probability. Therefore, we use the symbol sequence \( Z_{t} \) for the new definition of word distance measure \( \tilde{D} \):
\[
\tilde{D}(w_i, w_j) := \log P(Z_{t} | \hat{w}_i) - \log P(Z_{t} | \hat{w}_j),
\]  
where \( Z_{t} \) is defined by
\[
Z_{t} := \arg\max_{Z_{t}} P(Z_{t} | \hat{w}_i, X_{t}).
\]  
In the definition (9), the probability \( P(Z_{t} | \hat{w}_j) \) is computed with the scoring procedure (e.g., the forward algorithm, or the Viterbi algorithm). If the approximation \( E[\log P(y_t | \hat{w}_i)] \approx \log P(Z_{t} | \hat{w}_i) \) is warranted, the distance measure \( \tilde{D} \) has similar properties to the distance measure \( D \) of (4).

### 2.3. Normalization

The distance measure \( \tilde{D}(w_i, w_j) \) is highly dependent on the length of the symbol sequence \( Z_{t} \) generated for the word \( w_i \). It is important, therefore, to normalize the distance measure \( \tilde{D}(w_i, w_j) \). One approach is to normalize the term \( \log P(Z_{t} | \hat{w}_i) \) in the definition (9) by
\[
\log P(Z_{t} | \hat{w}_i) := \frac{\log P(Z_{t} | \hat{w}_i)}{T},
\]  
where \( T \) is the length of the sequence \( Z_{t} \). Another approach is by
\[
\log P(Z_{t} | \hat{w}_i) := \frac{\log P(Z_{t} | \hat{w}_i)}{\sum_{j=1}^{N} \log P(Z_{t} | \hat{w}_j)}.
\]  
such that \( \sum_{j=1}^{N} \log P(Z_{t} | \hat{w}_j) = 1 \). As the other approach, we propose the normalized distance measure as follows:
\[
\tilde{D}(w_i, w_j) := \text{Order}(w_i | \tilde{D}(w_i, w_j)),
\]  
where \( \text{Order}[] \) is the mapping to the order of word \( w_j \) according to the distance \( \tilde{D}(w_i, w_j) \). To sum up, we intend to normalize the distance \( \tilde{D} \) of (9) by the mapping into the integer \( 0, 1, \ldots, N - 1 \). By these normalization, we can evaluate all distance \( D(w_i, w_j) \), \( 1 \leq i \leq N, 1 \leq j \leq N \), based on the same criterion.

As the definition of word distance measure, Juang and Rabiner [6] have proposed a probabilistic distance measure for HMMs,
\[
D(\lambda_1, \lambda_2) = \lim_{T \to \infty} \frac{1}{T} \{ \log P(O_T | \lambda_1) - \log P(O_T | \lambda_2) \}.
\]
Furthermore, the modified distance measure
\[ D(\lambda_i, \lambda_j) = \frac{1}{N} \sum_{i=1}^{N} \left( \log P(A_{i,k} | \lambda_i) - \log P(A_{i,k} | \lambda_j) \right) \] (15)
is used by D’Orta, Ferretti, and Scari [7]. Here the symbol sequence \( A_T \) or \( A_k (k = 1, \ldots, N) \) is generated according to the state transition probability and the observation symbol probability of the model \( \lambda_i \). But they are randomly generated. This is a different point from our definition (9).

3. Word-node Graph

Based on the distance measure \( D(w_i, w_j) \), all words \( w_j (j = 1, \ldots, N) \) are distributed over a space with word \( w_i \) as the origin. It must be noted that no metric space is defined for words, because the axiom for metric is not satisfied by the distance measure \( D(w_i, w_j) \).

Now, we construct a graph keeping the structure of word distribution. The procedure is as follows:

1. **Initialization** : Define the node set
   \[ G_0 := \{ w_1, w_2, \ldots, w_N \} \] (16)
   where the nodes correspond to words. Set \( n = 0 \).

2. **Word Connection** : For the node set \( G_n \), connect the word pairs \( w_i \) and \( w_j \) satisfying
   \[ D(w_i, w_j) \leq r_s \quad \text{and} \quad D(w_j, w_i) \leq r_s \] (17)
   where \( r_s \) is a certain threshold. Then we get some connected graphs including simple node graphs.

3. **Representative Selection** : For each connected graph in step 2, select the node having maximal connection, as the representation. Next, remove these representative nodes and other nodes connected with the representatives, so that some subgraphs remain. Again, select the representative nodes from this subgraphs. This process is repeated until all nodes are removed. As a result, \( G_n \) is divided into representative nodes and non-representative nodes.

4. **Bipartite Graph Expansion** : Define \( G_{n+1} \) as the representative node set of \( G_n \). Set the directed path from the node of \( G_{n+1} \) to the node of \( G_n \) according to the connection in step 2. Also, set the directed path from the node of \( G_{n+1} \) to the self-same node of \( G_n \). Thus we get the bipartite graph, where all nodes of \( G_n \) are connected from at least one node of \( G_{n+1} \).

5. **Path Addition** : For all nodes \( w_i \) of \( G_n \), set the directed path from all nodes \( w_j \) of \( G_{n+1} \) satisfying
   \[ D(w_i, w_j) \leq r_t \] (18)
   where \( r_t \) is a certain threshold.

6. **Completion** : Replace \( n \) by \( n + 1 \). If the element number of \( G_n \) is less than a certain threshold, then halt. Otherwise, return to step 2.

We get \( G_0, G_1, \ldots, G_n \) in turn. The directed paths are connected from the nodes of \( G_{n+1} \) to that of \( G_n \). Step 2-5 are repeated using only the representative nodes from the current layer. After all, we obtain a layered, hierarchical, directed graph. All nodes correspond to words, and the leaves, the nodes of \( G_n \), correspond to the vocabulary. We call this graph a **word-node graph**.

From (17) and (18), it follows that
\[ D(w_i, w_j) \leq r_s \quad \text{or} \quad D(w_i, w_j) \leq r_t \] (19)
for all pairs of child node \( w_i \) and parent node \( w_j \) on the word-node graph. It means that the expectation \( E[\log P(y_i | \tilde{w}_i)] \) is as large as \( E[\log P(y_i | \tilde{w}_j)] \). Therefore, if the word \( w_j \) in the \( n \)-th layer has high probability \( P(w_j | y) \) for a given sequence \( y \), then we can hypothesize as follows:

- \( y \) is the symbol sequence corresponding to the word \( w_j \).
- \( y \) is the symbol sequence corresponding to the child word \( w_i \), connected from \( w_j \), in the \((n-1)\)-th layer.

Based on these hypotheses, we realize a recognition method using the word-node graph.

A vocabulary of words to be recognized is called a dictionary. And the combination of dictionary and the corresponding word-node graph is called a **graph-structured dictionary**.

4. New Search Algorithm

In this section, we present new search technique using a graph-structured dictionary. Define the dictionary
\[ W := \{ w_1, w_2, \ldots, w_N \} \] (20)
where \( w_i \) is word, and \( N \) is the vocabulary size. After building an HMM for each word \( w_i \), a word-node graph for the dictionary \( W \) is designed beforehand using only the HMM parameters. The bottom layer corresponds to the dictionary. If we have the word-node graph of \( L \)-layers, the search procedure is realized as follows:

1. **Initial Search** : Given a symbol sequence \( y \), calculate the probability \( P(w_j | y) \) for all words \( w_j \) in the \( L \)-th layer (i.e., the top layer). Store \( P(w_j | y) \), and set \( n = L \).

2. **Structural Search** : Select the \( m \) most probable words in the \( n \)-th layer, and calculate the probability \( P(w_j | y) \) for their children in the \((n-1)\)-th layer, where the probability calculation is omitted for the words that have already been examined. Store \( P(w_j | y) \).

3. **Recognition** : Set \( n \) by \( n - 1 \), and repeat the step 2 until the bottom layer is reached (i.e., \( n = 0 \)). Finally, select the most probable word as the recognition result from among the words having the probability.

The word candidates are effectively selected in each layer according to the word-node graph. Thus it is possible to restrict the search complexity, and to speed up the recognition.
process. Furthermore, the lexical tree for scoring is constructed not for all words of the dictionary, but successively for the word candidates in each layer. For convenience, step 1 is called an initial search, the repetition of step 2 is called a structural search, and \( m \) in step 2 is called a beam width.

4.1. Computational Quantity

We estimate roughly the number of words to be examined. Assume that, for the dictionary of vocabulary size \( N \), the word-node graph of \( L \)-layers is constructed as follows:

- The node number of \((n + 1)\)-th layer is half as many as that of \( n \)-th layer.
- Parent nodes in \((n + 1)\)-th layer are connected to 10 child-nodes in the \( n \)-th layer on the average.

Then the word number \( N_L \) for initial search is

\[
N_L = \left( \frac{1}{2} \right)^L N,
\]

and the word number \( N_S \) for structural search is

\[
N_S = (10 \times m) \times L,
\]

where \( m \) is beam width. For example, if \( N = 2000, L = 3, m = 10 \), then \( N_L + N_S = 250 + 300 = 550 \). The word duplication makes the total number of examined words less than \( N_L + N_S \). Moreover, it is easily verified that if \( N_L \) and \( m \) are fixed, then \( N_S \) is restricted to \( O(\log N) \). Thus, we see that our graph search is considerably effective for a large vocabulary word recognition.

5. Experimental Results

We compared the full search and our new search using a word-node graph. The 38 phoneme-based HMMs were used, where each HMM was a left-to-right model with no skipped states. The scoring was performed using the Viterbi algorithm. Our models were trained by 8 speakers' 1066 word utterances. As the test set, 4 different speakers' 1066 word utterances were used. The utterances were sampled at 12 kHz, and analyzed at 30-msec intervals. Table 1 shows the recognition accuracy and the number of words examined for the dictionary of 2029 words. The accuracy degraded according to the beam width, but the degradation was fairly small. With the beam width 15, the total number of examined words was restricted to 584 words. In this experiments, we used the normalized distance \( D \) of (13) for the design of word-node graph. This graph had 3-layers. And the initial search had 257 words.

Similarly, Table 2 shows the results for the dictionary of 14903 words. We used a word-node graph of 5-layers. The initial search had 445 words. Comparing Table 1 and Table 2, we notice that the total number of examined words increased by only about 400, although the vocabulary size dramatically increased from 2029 to 14903.

6. Conclusion

We have proposed a new search algorithm using a graph-structured dictionary. This is a word pre-selection technique. It is concluded that a distance measure defined by (9), the approximation of (4), is useful for word pre-selection. Moreover, the distance is calculated by the use of only model parameters. Therefore, we can easily design a word-node graph beforehand, based on the distance measure. Using this graph, we can significantly reduce the number of words to be examined, and obtain a recognition result at higher speed.

7. REFERENCES