SPEED IMPROVEMENT OF THE TIME-ASYNCHRONOUS ACOUSTIC FAST MATCH

M. Novak M. Picheny

IBM T.J. Watson Research Center
P. O. Box 218, Yorktown Heights, NY 10598, USA
(miroslav,picheny)@us.ibm.com

ABSTRACT

This paper describes an algorithm for improvement of the speed of a time-asynchronous fast match, which is a part of a stack-search based recognition system. This fast match uses a phonetic tree to represent the entire vocabulary of the recognizer. Evaluation of the tree (in a depth-first manner), can be done much more efficiently using the fact that under certain conditions, the results of branch evaluations can be used to approximate the scores of other branches of the tree.

1. INTRODUCTION

In this article an improvement of the existing fast match (FM) algorithm [1] used in the IBM recognition system [2] is described. The time-asynchronous FM system has several desirable features, such as efficiency, no requirement of a backward trace, and low memory requirements (thanks to depth-first search of the vocabulary tree). The most serious drawback of this algorithm is the fact that the vocabulary can only be organized into a phonetic tree, but not into a network, i.e. no merging of paths is allowed. This leads to a certain redundancy in the tree search, because equivalent parts of different phone sequences have to be evaluated repeatedly. The proposed method relies on certain assumptions, under which this redundancy can be significantly reduced.

At each iteration of the search algorithm, the FM is required to provide a list of word candidates for extension of the given hypothesis. Let’s say this hypothesis ends at time $t_0$ (the time information is actually provided in a form of pdf, called an end-time distribution but for simplicity we use the position of the pdf’s global maximum to denote the ending time). The FM will find a set of words matching reasonably well over a time region starting at $t_0$. Let the acoustic observation $O$ associated with this time region be a sequence of feature vectors $x = (x_{t_0} \ldots x_{t_k})$. Let score of a particular phonetic sequence $p = p_0 p_1 \ldots p_n$ be defined as:

$$S(p) = \sum_{t=t_0}^{t_{right}} \text{Prob}(x_{t_0} \ldots x_t|p).$$

(1)

We use simplified HMM based phone models, described in more detail in [1]. Then (1) is a summation of the forward pass probabilities of the Baum-Welsh algorithm:

$$S(p_n) = \sum_{t=t_0}^{t_{right}} a_{p,\text{end}}(t),$$

(2)

where $a_{p,\text{end}}$ is the forward probability of the last node of the whole HMM representing the path $p$. The interval $< t_{left} \ldots t_{right} >$ denotes the part of $< t_0 \ldots t_k >$ interval in which the probabilities are greater than zero after thresholding.

Having the score of a phone sequence (i.e. a particular path through the FM tree), we can define score of its last phone:

$$S(p_n|p_{n-1} \ldots p_0) = \frac{S(p_0 \ldots p_n)}{S(p_0 \ldots p_{n-1})}.$$  

(3)

Then it is possible to rewrite the path score in the following form:

$$S(p) = S(p_n|p_n-1 \ldots p_0)S(p_{n-1}|p_{n-2} \ldots p_0) \ldots S(p_0).$$

(4)

The score of a particular phone depends on the previous phones, but mainly on the immediately preceding one. We have found experimentally that in most cases the dependency can be limited to the preceding phone, so $S(p_n|p_{n-1}) \sim S(p_n|p_{n-1} \ldots p_0)$. Under this assumption, the score of the whole path can be obtained by combining it’s individual di-phone scores:

$$S(p) = S(p_n|p_{n-1})S(p_{n-1}|p_{n-2}) \ldots S(p_0).$$

(5)

It should be understood that the di-phone scores are computed over its corresponding time region $< t_{left} \ldots t_{right} >$. To be able to assure that, we need to associate the di-phone scores with the time region over which they are valid. As a result of the evaluation of one branch of the FM tree we obtain the end-time distribution $P_{t_n}(t_{end} = t_t|x)$. Since it strongly depends
on the acoustic observation \( x \), it is not possible to parameterize it by any known family of distributions. But it usually has one very distinct maximum located at certain time, called the most likely boundary \((t_{mlb})\). This time value can be used to uniquely identify the time region over which the di-phone score is valid. Another possibility is to use the expected value of \( t_{end} \) given the \( P_{\gamma x}(t_{end} = t_{i|x}) \). These two values quite often coincide, since the peak of the distribution is very sharp (the expected value is computed in linear domain). The choice really depends on the actual implementation of the HMM evaluation and the platform used. A very efficient algorithm for the \( t_{mlb} \) computation is discussed later in this article.

In figure (1) an example of a three-word FM tree evaluation is shown, with the corresponding end-time distributions. Since the probability axis has a logarithmic scale, the scores can be measured as the distance between \( t_{mlb} \) points along the y-axis. As can be seen, the scores of the the first common phone \( Y \) is quite different on each path, but the scores of the following phones are equivalent.

### 2. ALGORITHM DESCRIPTION

The proposed method enhances the existing FM search by incorporating a cache for the di-phone scores. The FM is called at certain time \( t_0 \) to find all possible extension of the current word sequence hypothesis. The FM tree is traversed in the depth-first manner, in each step one branch (associated with one phone) of the tree is evaluated. The result of this evaluation is the end-time distribution and score \( S_{x,y} \). The score of the whole path can be obtained by multiplication of all of its phonetic scores. In the new algorithm, before the phone computation is preformed, the cache is checked for existing scores. If the score is found, together with its corresponding end time \( t_{mlb} \), it is used to extend the current path. Otherwise, the regular phone evaluation is performed and the results are inserted back to the cache. Obviously storing the whole end-time distribution is not feasible. But that is only needed if the score of the next phone is not found in the cache. In such case the algorithm has to back off to obtain end-time distributions of all phones in the path. The following table summarizes the observed speedup potential in terms of FM tree nodes:

<table>
<thead>
<tr>
<th></th>
<th>Total visited</th>
<th></th>
<th>Computed</th>
<th></th>
<th>Back off</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8817</td>
<td></td>
<td>2885</td>
<td>32.7%</td>
<td>1164</td>
</tr>
</tbody>
</table>

The number of computed nodes includes the instances when the score either was not found in the cache or it had to be recomputed during back-off step.

### 3. CACHE ORGANIZATION

To use the cache memory efficiently, use of some form of a hash-table algorithm is desirable to convert the 3-dimensional vector \((p_{n-1}, p_n, t_{mlb}(p_{n-1}))\) into a unique key. Rather than using any of the standard hashing algorithms, we decided to implement our own algorithm which tries to maximally utilize our knowledge about the nature of the data. The couple of \((p_{n-1}, p_n)\) can be mapped to a single value using the fact that given the lexicon only certain di-phone combinations are possible, and on top of that, some of those are quite infrequent. In our implementations, the original space of \(52^2 = 2704\) di-phones was reduced to 1441. If time is measured continously form the beginning of the utterance, at the certain point the of search, the values of \( t_{mlb} \) can be expected to lie in a relatively small interval. Let's assume this interval be 100cs (100 feature vectors), then the size of a direct lookup table would be \(100 \times 14441\). This can be further reduced by quantizing the time value. We have found that dividing the \( t_{mlb} \) by 4 doesn't have any effect on the final accuracy of the system and actually slightly increases the cache hit rate. The individual cells of the table are automatically recycled as the search makes progress. Each cell contains the following information:

\[
\begin{align*}
S(p_n) & \quad \text{score of the phone} \\
t_{mlb}(p_n) & \quad \text{ending time of the phone} \\
t_{mlb}(p_{n-1}) & \quad \text{ending time of the previous phone}
\end{align*}
\]

Since the asynchronous search algorithm used in our recognizer practically always moves forward in time when a new search path is being extended, it is possible to declare a time boundary \( t_{ref} \) which marks the part of the utterance we have no interest in anymore. The cache is addressed by the di-phone index and quantized value of \( t_{mlb}(p_{n-1}) \). If this time value agrees with the one stored in the cell, a cache hit is declared. If not, the cell is either reused (when \( t_{mlb}(p_{n-1}) \) stored in the cell \( < t_{ref} \)) or a hashing conflict occurs. We have measured that the conflict occurs in less than 0.5% of all cases. We have also measured that the hash-table utilization is more that 70%.

Figure (2) shows a histogram of \( \log_{10}(S(p_n)) \) and the score error caused by using the cached values instead of the computed ones. The next figure (3) shows a histogram of errors of the \( t_{mlb} \). Note that in both cases the counts has to be on a logarithmic scale to discern the values of the histograms.

### 4. END TIME COMPUTATION

The estimation of the most likely ending time \( (t_{mlb}) \) represents an overhead of this method and needs to be computed very efficiently. A simple linear search for the maximum is not fast enough. We tried to
incorporate the search into the HMM evaluation procedure. The expected value of $t_{end}$ can be expressed as:

$$E[t_{end}] = \frac{1}{S(p_n)} \sum_{t_{right}} t \alpha_{p_n, end}(t)$$

(6)

This summation can be done very efficiently. The following table can help to visualize why:

$$E[t'_{end}] = \sum_{t'_{right}} \sum_{\tau=0}^{t'_{right}} \sum_{\tau=0}^{t_{left}} \alpha_{p_n, end}(\tau)$$

(7)

where $t'$ is a reversed and shifted version of $t$.

<table>
<thead>
<tr>
<th>$t$</th>
<th>$\sum_{\tau=0}^{t} \alpha_{p_n, end}(\tau)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\alpha(0)$</td>
</tr>
<tr>
<td>1</td>
<td>$\alpha(0)+\alpha(1)$</td>
</tr>
<tr>
<td>2</td>
<td>$3\alpha(0)+2\alpha(1)+\alpha(2)$</td>
</tr>
</tbody>
</table>

Since the value of the inner sum is available at each time frame, the actual overhead is only one addition per frame.

Figure 1: Example of a FM tree

This method works well on platforms with efficient multiply/add instructions with double float precisions to work directly in the linear domain. On some platforms, the fixed point operations are more efficient, but the dynamic range requires to work in the logarithmic domain. In such case, the additions become expensive and can be replaced by $\max$ of two scores (turning the Baum-Welsh forward pass into a Viterbi score computation). Then it is actually better to use the value of $t_{mlb}$, since it can be found at little expense as a part of the max operation.

5. RESULTS

The algorithm was tested on a research prototype of the IBM speech recognition system [3]. The system uses different acoustic models for sub-phonetic units in different contexts. These instances of context dependent classes are identified by growing a decision tree from the training data. The acoustic feature vectors that characterize the training data at the leaves are modeled by a mixture of Gaussians pdf’s, with diagonal covariance matrices (a total of 32626 Gaussians were used). As far as the output distributions on the state transitions of the model are concerned, rather than expressing the output distribution directly in terms of the feature vector, the IBM system expresses it in terms of the rank of the leaf [3]. The acoustic front end uses a FFT based filter bank followed by cepstral rotation. Frame energy and dynamic parameters ($\Delta + \Delta \Delta$) were added to each feature vector. Sentence based cepstra mean normalization was used.

The test set consists of 61 sentences each read by 10 speakers. The decoder runs in a speaker independent mode. The following table shows the recognition error rates and achieved speed improvements.

<table>
<thead>
<tr>
<th></th>
<th>baseline</th>
<th>new</th>
<th>impr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WER</td>
<td>14.74%</td>
<td>14.80%</td>
<td>-0.004%</td>
</tr>
<tr>
<td>FM time[s]</td>
<td>1247</td>
<td>780</td>
<td>37.4%</td>
</tr>
<tr>
<td>Total time[s]</td>
<td>3229</td>
<td>2807</td>
<td>12.9%</td>
</tr>
</tbody>
</table>
6. CONCLUSION

We have presented a method speeding up the asynchronous fast match algorithm. This method causes a negligible degradation of the recognition accuracy. While there is a certain overhead introduced, the overall speed improvement is still substantial.

7. REFERENCES

