Proceedings of the Institute of Acoustics

A DYNAMIC LEVEL BUILDING ALGORITHM FOR LARGE VOCABULARY LEXICAL ACCESS*

P. Nowell

Speech Research Unit, Defence Research Agency, St. Andrews Rd, Malvern, WR14 3PS, England.

1 INTRODUCTION

The concept of dynamic programming (DP) was introduced by R. Bellman [1] in 1957 and was first applied to the problem of time aligning speech signals by T. Vintsyuk [2] in 1968. Since then a number of efficient dynamic time warping (DTW) algorithms have been devised which use dynamic programming techniques to find the optimal time alignment of reference templates with paramaterised representations of isolated words [3] as well as connected speech [4] and [5], [6]. Dynamic programming techniques have also been found to provide a robust and reliable means of lexical access [7]. The apparent similarity between the processes which are involved in dynamic time warping of essentially continuous speech signals and those which are required to align naturally discrete phoneme sequences suggest that the efficient DTW algorithms might also be applicable for use in lexical access.

The two most widely used connected word DTW algorithms are known as the level building algorithm [4] and the more efficient, one pass algorithm [5], [6]. It has been shown [8] that both algorithms are in fact functionally equivalent. These algorithms will produce identical results when used to align speech signals provided that the maximum number of levels (i.e. maximum number of words) in the level building algorithm is sufficiently large. However, when we come to use these algorithms for lexical access we discover that these algorithms are no longer functionally identical. There are problems which arise in the one pass algorithm which do not arise in the level building algorithm. In this paper we will describe these problems along with a solution which combines the features of both algorithms in order to overcome them. This will lead to an algorithm which we call the dynamic level building algorithm.

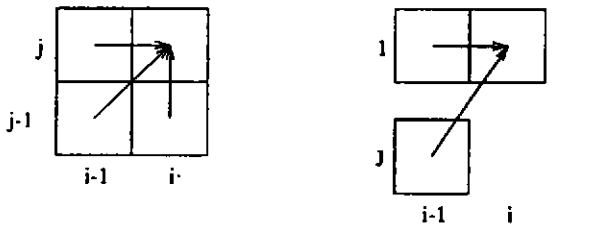
2 DTW AND DISCRETE SEQUENCE ALIGNMENT

Classical DTW algorithms operate by selectively compressing and expanding portions of essentially continuous speech signals in order to obtain the optimal time alignment between an input signal and the reference template(s). The compression and expansion processes are used to time align the speech signals are similar to the insertion and deletion processes that are needed to align naturally discrete sequences such as phoneme transcriptions. The similarities between these two

*This paper describes research which was carried out as part of a PhD project at the Dept. of Artificial Intelligence, Univ. of Edinburgh, Edinburgh, Scotland.

processes suggest that it may be possible to adapt classical DTW algorithms for use in lexical access.

In practice the compression and expansion of speech signals can be achieved by selectively deleting, inserting, and substituting sequences of parameterised speech vectors. Dynamic programming techniques allow the globally optimal alignment to be constructed by gradually extending the locally optimal alignments. With each new input frame i the alignment between the $J(k)$ frames of each template k and the input is extended according to a set of path constraints (fig. 1). The path constraints are used to enforce continuity constraints and also ensure that paths are always extended from bottom to top and left to right.



(a) Within Word Constraints (b) Between Word Constraints

Figure 1: Symmetrical DP Path Constraints

Connected word DTW algorithms use different path constraints when propagating paths within templates (fig. 1(a)) than when propagating paths between templates (fig. 1(b)). The between word path constraints are more restrictive than those which are used within words. They do not permit new paths to grid point $(k, i, 1)$ to be propagated from the last grid point $(k, i, J(k))$ of the lowest scoring template k whose alignment also ends at the same point i. Such path constraints ensure that the first point of a new alignment within a template is always reached by substituting the first frame of the template. These constraints prevent compressions from following expansions and also prevent templates from being infinitely compressed i.e. from having alignments which start and end at the same point in time.

The fundamental difference between the level building and one pass algorithms occurs in the propagation of partial alignments between word boundaries. The level building algorithm propagates partial alignments from the optimal template in the preceding level whilst the one pass algorithm propagates paths between templates at the same level. As we shall see, this leads to problems for the one pass algorithm when we come to try and use it for lexical access.

$$D^*(c,k,s,j) = \min_{i=1\ldots I(s)} \begin{cases} D(c,k,t_s(s),j) & +d(i,-) & +ac(i) & \text{;Delete input} \\ D(c,k,t_s(s),j-1) & +d(i,j) & +ac(i) & \text{;Substitute} \\ D(c,k,t_e(s),j-1) & +d(-,j) & & \text{;Delete template} \end{cases}$$

for $l = 1 \ldots L$
  for $s = 1 \ldots S$

1) Between Word Syntactic Constraints

for $c = 1 \ldots C$
  for $k = 1 \ldots K(c)$
    $D(c,k,t_e(s),0) = \min_{c_s=1\ldots C_s} T_{l-1}(t_e(s),c_s)$

2) Within Word DP Alignment

for $c = 1 \ldots C$
  for $k = 1 \ldots K(c)$
    for $j = 1 \ldots J(k)$
      $D(c,k,t_e(s),j) = D^*(c,k,s,j)$
  $T_l(t_e(s),c) = \text{argmin}\{D(c,k,t_e(s),J(k))\}$

Figure 2: A General DP Alignment Algorithm

3 DP LEXICAL ACCESS

The input to our dynamic programming algorithm consists of lattices of phoneme segments which were produced by an acoustic front end (AFE) developed at CSTR[1] [9]. The input lattices consist of S phoneme segments each of which contains I different phonemes. Although the lattices are well-structured in the sense that each phoneme i in a segment s has the same starting time $t_s(s)$ and the same finishing time $t_e(s)$ this is not essential. The phonemes in each segment are ranked according to a likelihood measure $ac(i)$ of their presence as calculated by the AFE. An optional chart parser can be used to enforce grammatical constraints in the form of context-free phrase structure grammar rules [11]. In order to facilitate parsing the lexicon is partioned into C syntactic categories (parts of speech) according to the terminal categories of the grammar. Each category c in the lexicon contains $K(c)$ templates and each template k contains $J(k)$ frames which contain the phonemic labels for a single citation form pronunciation of the word.

The algorithm outlined in fig. 2 is used to implement either a one pass or level building DP algorithm, with or without syntactic constraints. The actual algorithm depends upon how paths are propagated across word boundaries i.e. the between word syntactic constraints. In

[1]Centre for Speech Technology Research, University of Edinburgh, Edinburgh, Scotland.

the level building algorithm paths are propagated from templates in the preceding $(l-1)$ layer whereas in the one pass algorithm paths are propagated from templates in the same $(L=1)$ layer. Symmetrical path constraints are used (fig. 1a) which permit an unlimited number of consecutive insertions and deletions. However, unlike traditional DTW algorithms we do not distinguish between within word and between word path constraints. Insertions (i.e. deletion of input phonemes) and deletions (i.e. deletion of template phonemes) are in fact implemented as substitutions with the null pseudo-phoneme /-/ which has zero duration.

The dynamic programming algorithm obtains the optimal (lowest scoring) alignment on the basis of the scores provided by the acoustic front end and a matrix $d(i,j)$ of substitution penalties which are trained using an iterative maximum likelihood training (ML) procedure. The substitution penalties are recursively re-estimated using the ML training procedure which (locally) maximises the likelihood of the alignments between a set of training sentences and the corresponding phoneme lattices produced by the acoustic front end.

4 WORD INITIAL DELETIONS

The between word path constraints in traditional DTW algorithms (fig. 1(b)) are more restrictive than the within word path constraints (fig. 1(a)) which do not allow word initial deletions. We can not separate these two cases since we need to have word initial deletions so that correct alignments such as (@.-)(n.n)(d.-) for 'and' and (dh.-)(@.@) or (dh.-)(@.-) for 'the' are obtainable. Unfortunately the introduction of word initial deletions results in a number of problems for the one-pass DP algorithm which would not be encountered by a level building algorithm. The reasons for the difficulties can be seen by examining the way in which paths are extended across word boundaries in the level building and one pass algorithms as indicated by the arrows in fig. 3.

The level building algorithm constructs the optimal template alignment level by level. The optimal templates in one level are determined before work begins on the next level. The paths which are obtained via word initial deletions (corresponding to the vertical arrows in fig. 3) are extended from the paths in the final frame of templates in the preceding level. The word initial paths in the new template are therefore extended from paths in the preceding level that finish *at the same time* as the new paths originate. This is not a problem for the level-building algorithm since the best scoring templates containing paths which finish at this time will already have been determined and will be recorded in the previous level.

The one-pass algorithm effectively collapses the numerous levels of the level-building algorithm into a single level (fig. 3b). The paths which are obtained via word-initial deletions (vertical arrows) are now extended from the best scoring templates in the same level. This leads to a paradox. We cannot determine the optimal template until we have calculated scores for each of the $j=0\ldots J(k)$ reference frames. However, we cannot calculate score for the initial frame $j=0$ and subsequent frames until we know the score of the optimal template since the first substitution of the optimal alignment may consist of a word initial deletion from this template.

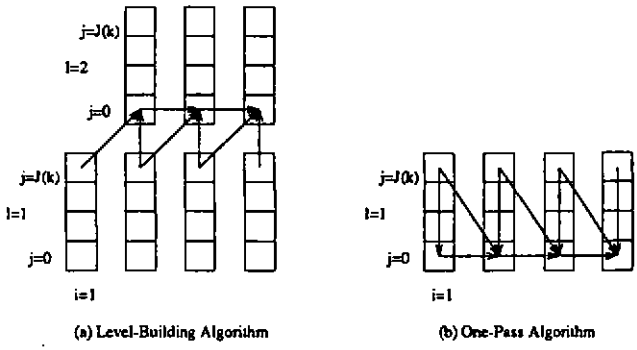


Figure 3: Propagation of paths between templates in the level building and one pass algorithms

4.1 A PARTIAL SOLUTION

The key to solving this problem is to observe that the path scores increase monotonically since the individual substitution penalties (negative log. probabilities) are by definition positive. This means that we can determine the optimal template alignments in two passes.

1. In the first pass word initial deletions are not used. The between word alignments are propagated using the standard, restrictive, path constraints as shown in in fig. 1b.
2. The optimal template having the lowest scoring alignment is determined. This template is then used to propagate paths in a second pass which implements the missing word initial deletions.

Note that the second pass will never affect the initial choice of the lowest scoring template which would in turn invalidate the scores of the recently calculated word initial deletions. If all of the phonemes in a template were to be deleted the score of the path would be equal to the score of the previously determined optimal template (from which the path originated) plus the sum of the individual deletion penalties. These alignment scores must increase monotonically since the substitution penalties are always positive. The new alignment score of a completely deleted template therefore cannot be lower than the score of the previously determined optimal template. This also means that a completely deleted template will never be part of the optimal alignment since there will always be at least one other template that has a lower alignment score.

5 DYNAMIC LEVEL BUILDING

When we come to generate the N-Best alignments [10], or enforce grammatical constraints [11] it is possible for a completely deleted template to form part of an N-Best or syntactically constrained

$$D^*(c,k,s,j) = \min \begin{cases} D(c,k,t_e(s),j) & \text{;Original path} \\ D(c,k,t_e(s),j-1) \quad +d(-,j) & \text{;Word initial deletion} \end{cases}$$

do

1) Between Word Syntactic Constraints

for $c = 1 \ldots C$
  for $k = 1 \ldots K(c)$
    $D(c,k,t_e(s),0) = \min_{c_s=1 \ldots C_s} T_{l-1}(t_e(s),c_s)$

2) Within Word DP Alignment

for $c = 1 \ldots C$
  for $k = 1 \ldots K(c)$
    for $j = 1 \ldots J(k)$
      $D(c,k,t_e(s),j) = D^*(c,k,s,j)$
  $T_l(t_e(s),c) = \text{argmin}\{D(c,k,t_e(s),J(k))\}$

while $T_l \neq T_{l-1}$

Figure 4: The second stage *dynamic level building* algorithm

alignment. When this occurs we will we have to update our initial choice for the set c_s of lowest scoring, syntactically valid, templates. Unfortunately, this will also invalidate the previously calculated scores for the word-initial deletions. We will therefore have to go back and recompute word initial deletions which originated from templates which may no longer be among the initial set or which may have lower scores if they were to originate from one of the newly deleted templates. This may in turn cause further changes to set of N-best or syntactically valid templates.

The solution to this more complex problem is an extension of that described previously. As before word-initial deletions are applied after the main DP pass.However the scores of templates that have been completely deleted must now be compared against the initial choice of the lowest scoring templates. If one or more of the newly deleted templates has a lower score than any of those in the initial set then a new 'level' of backtracking information is created which records the backtracking pointers for the newly deleted template(s). The second word-initial deletion pass is then repeated using deletions which extend paths from the final frames of the newly deleted templates in the previous level rather than from the final frames of templates in the initial set. In this case the propagation of paths across word boundaries mirrors that of the level building algorithm.

A number of levels of template backtracking information are now be stored, one for each set of templates that have alignments which finish at the same point in time (fig. 5). Each successful

iteration of dynamic level-building will result in the creation of a new level of backtracking information. The backtracking pointers for these completely deleted templates refer to information about templates in the preceding level whose optimal alignments also ended at the same time. The dynamic level building step is always executed once to allow for an arbitrary number of word initial deletions and is then repeated indefinitely whilst the process results in creation of new, lower scoring, completely deleted templates. Fortunately, it is rare in practice for even one template to be completely deleted.

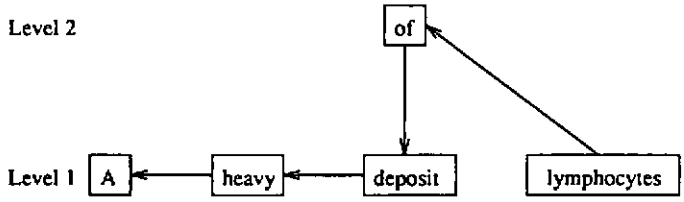


Figure 5: The storage of backtracking pointers in the dynamic level building algorithm

If the same word template appears consecutively in the optimal alignment the dynamic level building stage will overwrite the original paths and phoneme back-pointers with deletions. Although it would be possible to backtrack through the phoneme substitutions in the last, completely deleted template, it would not be possible to backtrack through the substitutions of the previous occurrence of the template since the necessary pointers will have been overwritten. Fortunately the likelihood of obtaining two consecutive alignments of the same template in an optimal alignment are very small, especially when syntactic constraints are in force and we have never encountered this problem in practice.

The dynamic level building preserves the relative advantages of the one pass algorithm over the alternative level building algorithm. The additional levels of backtracking information are dynamically created as and when they are required so that computational and storage overheads are kept to a minimum. There is also no need to specify a maximum number of levels. The second word initial deletion stage is also much quicker than the first DP stage since the only substitutions which are used are deletions whereas the first DP stage must consider deletions, substitutions, and insertions.

References

[1] R. Bellman. *Dynamic Programming.* Princeton Univ. Press. 1957.

[2] T.K. Vintsyuk. Speech discrimination by dynamic programming. *Kybernetika.* Vol. 4, 1968.

[3] H. Sakoe and S. Chiba. Dynamic programming algorithm optimisation for spoken word recognition. *IEEE Trans. Acoustics, Speech and Signal Proc.*, ASSP-26, 1978.

[4] C. S. Myers and L. R. Rabiner. A level building dynamic time warping algorithm for connected speech recognition. *IEEE Trans. Acoustics, Speech and Signal Proc.*, ASSP-29, 1981.

[5] J.S. Bridle, M.D. Brown, and R.M. Chamberlain. An algorithm for connected word recognition. In *Proc. IEEE Conf. on Acoustics, Speech, and Signal Proc.*, 1982.

[6] H. Ney. The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Trans. Acoustics, Speech and Signal Proc.*, ASSP-32, 1984.

[7] Henry S. Thompson, David McKelvie, and Fergus McInnes. Robust lexical access for continuous speech using dynamic time warping and finite-state transducers. In *Eurospeech 89 conference proceedings*, 1989.

[8] C. Godin and P. Lockwood. Dtw schemes for continuous speech recognition: a unified view. *Computer Speech and Language Journal*, 1989.

[9] F.R. McInnes, Y. Ariki, and A.A. Wrench. Enhancement and optimisation of a speech recognition front end based on hidden markov models. In *Eurospeech 89 conference proceedings*, 1989.

[10] P. Nowell. An efficient implementation of the n-best algorithm for lexical access. In *2nd European Conference on Speech and Communications Technology, Genoa, Italy*, 1991.

[11] P. Nowell. *Robust Lexical Access using Context Sensitive Dynamic Programming and Macro-Substitutions.* PhD thesis, University of Edinburgh, 1991.

© British Crown Copyright 1992