SENTENCE-HYPOTHESES GENERATION IN A CONTINUOUS-SPEECH RECOGNITION SYSTEM

Volker Steinbiss

Philips GmbH Forschungslaboratorium Hamburg, D-2000 Hamburg 54, P.O. Box 540840, FRG

ABSTRACT: In this paper, the dynamic-programming algorithm for continuous-speech recognition is modified in order to obtain a top-N sentence-hypotheses list instead of the usual one sentence only. The theoretical basis of this extension is a generalization of Bellman’s principle of optimality. Due to the computational complexity of the new algorithm, a sub-optimal variant is proposed, and experimental results within the SPICOS system are presented.

INTRODUCTION

The output of a continuous-speech recognition system using Viterbi decoding is the word sequence associated with the most likely hidden Markov model state sequence. In the continuous-speech database query and answering system SPICOS, this word sequence is passed to a language-interpretation module. Subsequently, data-base query evaluation, answer generation and speech synthesis have to be carried out in order to produce a spoken answer.

If, in the case of a speech recognition error, the recognized sentence cannot be parsed or interpreted, the system may have to stop at this point as no further information is available from the recognition module. So, it is desirable that the speech-recognition module does not pass one sentence only but a list of sentences instead which are most likely to have been uttered. For the level-building algorithm, an approach to the top-N decoding problem has been proposed in [5].

The subject of this paper is to modify the one-stage dynamic-programming search algorithm in order to produce a top-N list of sentence hypotheses. The memory and computation-time requirements of the exact approach seem to make it use in a system rather difficult. Therefore, in chapter 2, a sub-optimal approach with only moderate consumption of storage and processing time is described. The performance of this approach is subject to the experiments discussed in section 3.

1 A GENERALIZATION OF BELLMAN’S PRINCIPLE OF OPTIMALITY TO TOP-N WORD SEQUENCES

1.1 Bellman’s Principle of Optimality: The Optimal State Sequence

Viterbi decoding in connected-speech recognition solves the following problem: Given a hidden Markov model (HMM) ([1], [6]), find the state sequence which most probably has produced the observed input Y = y(1), ..., y(T). The word sequence W associated with this state sequence might be sub-optimal in the sense that it does not maximize the likelihood p(W,Y) ([1]); nevertheless good results have been obtained with Viterbi decoding, and it is also used within the SPICOS system (cf. section 3).

The problem can be reformulated as to find a state sequence (also called ‘path’) which minimizes an additive cost function:

Problem: Given a HMM (cf. section 1.2), find the state sequence s = s(0), ..., s(T) to minimize

where c(i,j) are the costs of moving from state i at time t-1 to state j at time t. This cost function is defined in the underlying model: In the HMM approach, it is the negative logarithm of the probability of going from state i at time t-1 to state j at time t and of producing the output y(t). As in the dynamic time warping (DTW) approach, it might just as well be some distance between a reference vector and the observed vector y(t), plus a time distortion penalty. We are not limited to either of these interpretations but would like to describe, as generally as possible, the dynamic-programming (DP) recursions (this section) as well as their generalization to generating top-N hypotheses lists (sections 1.2. and 1.3.).

In order to find the optimal state sequence through S states there is no need to evaluate all of the \( S^T \) possible paths. Indeed, Bellman’s simple but powerful principle of optimality ([2]) states that if the optimal state sequence from s(0) to s(T) passes through state s(t) at time t, then it includes, as a portion of it, the optimal state sequence from s(0) to s(t). In other words, a non-optimal partial path can never be part of the global optimal path.

Denote by \( P(t,s) = P(s(0), ..., s(t)) \) the optimal path from s(0) to state s = s(t) (or an optimal path; for conciseness, we will not consider non-uniqueness here nor do we define the case when there is no path to state s = s(t) ('infinite costs'), and denote its accumulated costs by \( C(t,s) \). The global optimal path \( P(T,s_f) \) from beginning state s_f to final state s_f and its accumulated costs \( C(T,s_f) \) can be calculated recursively by

\[
C(0,s) = \begin{cases} 0 & : s = s_f \\ \infty & : \text{else} \end{cases} \quad (1)
\]

\[
P(0,s) = \begin{cases} s_f & : s = s_f \\ \text{empty} & : \text{else} \end{cases} \quad (2)
\]

Recursion over \( t = 1, ..., T \):

\[
C(t,s) = \min_s \left( C(t-1,s') + c(s',s) \right) \quad (3)
\]

\[
P(t,s) = P(t-1,s')s \quad \text{is the state sequence obtained by concatenating} \quad P(t-1,s') \quad \text{with} \quad s \quad \text{where} \quad s' \quad \text{is the predecessor state that minimizes} \quad (3).
\]

Fig. 1 illustrates how the optimal path \( P(T,s_f) \) is obtained by successively following the minimizing arguments \( s' \).

Note that, in the general case, the complexity of calculating \( C(T,s_f) \) drops from \( S^{T+1} \) to \( S \cdot T \).

1.2 Top-N State Sequences

The optimality principle is generalized from seeking the optimal state sequence to seeking the top-N state sequences simply as follows: If, for \( n = 1, ..., N \), the n-th best state sequence from s(0) to s(T) passes through state s(t) at time t, then it includes, as a portion of it, the m-th best (with some \( m \in \{1, ..., n\} \)) state sequence from s(0) to s(t). In other words, a partial path not among local top-n can never be part of the global n-th best path.

First European Conference on Speech Communication and Technology
EUROSPEECH '89
Paris, France, September 27-29, 1989

EUROSPEECH '89, Paris, France, September 1989

2051
Let $P_n(t,s)$ denote the $n$-th best path from $s_0 = s(0)$ to $s = s(t)$ and let $C_n(t,s)$ denote its accumulated costs. Then the recursion is:

**Initialisation:**

\[
C_n(t,s) = \begin{cases} 0 & : n = 1 \text{ and } s = s_B \\ \infty & : \text{else} \end{cases}
\]

**Recursion over $t = 1, \ldots , T$:**

\[
P_n(t,s) = P_{n'}(t-1,s')s, \text{ where } n',s' \text{ minimize (9).}
\]

(The minimum is taken over those state sequences whose associated word sequence is not yet in the local top-(n-1) list.)

1.3 Top-N Best Word Sequences

A top-N list of state sequences is not yet what we want - it might result from different time alignments of only one word sequence. In order to get a list of top-N word sequences one must, during recombination, take into account for each state sequence $P$ the word sequence $W(P)$ associated with it. Accordingly, while the initialisation remains as (5) and (6), the recursion formulae have to be changed as follows:

Recursion over $t = 1, \ldots , T$.

Let, for $n = 1, \ldots , N$,

\[
C_n(t,s) = \min_{n',s'} \left( C_{n'}(t-1,s') + c(t,s',s) \right) \quad \text{(9)}
\]

\[
W(P_{n'}(t-1,s')s) \neq W(P_{n'}(t,s)) \text{ for all } m \neq n.
\]

(The minimum is taken over those state sequences whose associated word sequence is not yet in the local top-(n-1) list.)

Eventually, the same optimal state sequence $P_\star(T,s_P)$ and word sequence $W_\star := W(P_\star(T,s_P))$ are obtained as in the preceding sections. In contrast to the preceding section, $P_\star(T,s_P)$ here is the optimal path among all paths with an associated word sequence different from $W_\star$, etc.

2 A SUB-OPTIMAL VARIANT OF TOP-N DYNAMIC PROGRAMMING

Using the recursion formulae (5), (6), (9) and (10), the usual onestage dynamic-programming algorithm (cf. [12], [3], [7]) can be modified in order to obtain a top-N sentences hypotheses list instead of one sentence only. For an efficient implementation, the following seems to be important:

- A fast access to the word-sequence list associated with a point in the search space. This can be realized by constructing a tree containing all partial word sequences so far seen and handling only its child nodes within the recombination.
- A fast sorting or merging procedure for the recombination of top-N lists.

As the search space is rather big - though using a pruning strategy, in SPiCOS several thousand points have to be processed for every centisecond of speech - the extra effort for getting top-N sentence hypotheses instead of one sentence seems to be rather high. This relates to the computational costs for the recombination as well as to the storage of top-N lists. Therefore, we propose a sub-optimal variant of top-N DP by restricting the extended recombination (formulae (9) and (10)) to the syntax level and letting the in-word recombinations be the standard ones (formulae (3) and (4)). The effects of this clearly sub-optimal strategy are discussed below. Its advantage with respect to the exact top-N DP search is the reduced computational and storage complexity, because most interactions are in the word interiors. Experimental results are presented in section 3.

The proposed approach ignores the fact of in-word recombinations: It does not obtain the exact solution, i.e. the best path giving the n-th best word sequence, if and only if this path recombines somewhere inside a word with a better scored path. Whereas on the syntax level all necessary information about the (locally) top-n word sequences is stored, only best decisions are regarded inside words (on the state level). That means the algorithm finds a sub-optimal solution only.

To see in more detail what happens in such a case, regard fig. 3 as an example which shows a part of the search space. Inside any of the words $u$, $v$, $w$, the HMM is left-to-right; from the word ending states, every word beginning state can be reached. Imagine that $uw$ and $vw$ are the best and second best word sequences, resp. Three paths, each of which optimal in a certain sense, are subjects of our interest:

- The global optimal path $P_\star$ (through word sequence $uw$)
- The best path $P_2$ through the second best word sequence $vw$. It is lost (i.e. unavailable for trace-back) because it recombines inside word $w$ with the better scored path $P_1$.
The best path through $w$ which is available for trace-back is
- path $P_3$, which is the best path through $w$ satisfying the additional
  requirement that the word boundary between $v$ and $w$ coincides
  with the word boundary between $u$ and $w$.

  Generally, $P_3$ will have a better score than $P_3$, that is, the path $P_3$
found is only sub-optimal. On the other hand, if $P_2$ recombines with
$P_1$ inside $w$, there is some indication that $P_2$ and $P_3$ are relatively close
to each other anyway and that so are their costs. This can be taken as
a justification for using this sub-optimal decoding strategy in a sys­
tem. The experimental results of the following chapter show indeed
that although the sub-optimality is visible, the proposed approach
solves the top-N decoding problem with sufficient accuracy.

Fig. 3: Illustration of the approximation error. The best path $P_2$ for
word string $vw$ recombines with the globally best path $P_1$ (word string
$uw$) inside word $w$ and is thus unavailable for trace-back. Only the path
$P_3$ through $uw$ which passes the syntax node at the same time as the
path through $vw$ (i.e., the word boundaries are reached at the same
time) can be recovered by trace-back.

3 EXPERIMENTAL RESULTS

3.1 The Recognition System Environment

Experimental tests were performed within the SPICOS system,
with 'SPICOS' standing for 'Siemens-Philips-IPO continuous-speech
recognition and understanding system'. Within this joint project a
man-machine dialogue system has been developed which provides
voice access in German to a database containing information about
the project itself. The SPICOS system consists of modules for speaker­
dependent continuous-speech recognition, language interpretation,
database-query evaluation, answer generation, and speech synthesis.
For details, cf. [8], [9], [10], [11].

The phoneme-based large-vocabulary speech-recognition mod­
ule of SPICOS as used for the tests described below can be character­
ized as follows:
- Office environment, close-talking microphone.
- Preprocessing:
  - Sampling rate 16 kHz
  - 30 cepstral smoothed spectral intensities in logarithmic units,
    normalized with respect to average intensity, plus intensity
  - Additionally, first and second differences of these.

Acoustic-phonetics:
- Standard pronunciation dictionary (one pronunciation per word)
- 44 context-independent phonemes
- Continuous mixture-density distributions.

Search:
- Data-driven one-stage DP search, modified according to chapter
2 in order to obtain (sub-optimal) top-N sentence hypotheses lists.

Language model:
- Recognition vocabulary comprising 917 words
- Either finite state grammar (test set perplexity [4] 85) or stochastic
  bigram language model (test set perplexity 124).

Training and test data:
- 2 times 100 phonetically balanced training sentences (7 minutes
  of speech)
- 200 test sentences (data-base queries)
- Small overlap only of training and recognition vocabularies.

3.2 Recognition Results

For each of four speakers (two male, two female), recognition
tests have been performed on 200 test sentences. Table 1 shows, for
$N$ from 1 to 10 and the two language models, how often the spoken
sentence is included in the top-N sentence hypotheses list. For the
finite-state language model, more than one half of the spoken sen­tences
which were not correctly recognized (i.e., not in position 1) was found among top 4.
On the other hand, one third was not even among
top 12. For the bigram language model, one third of the incorrectly
recognized sentences can be found among top 5. For completeness,
the top-N error rates are given in table 2, defined as deletions + insertions + substitutes of (for each spoken sentence) the best recogni­tion in the top-N list, divided by the number of spoken words. Both
top-N sentence recognition rate and top-N error rate show what at best
could be achieved by post-processing the sentence-hypotheses lists
with a powerful parser; in some cases, however, not even human
knowledge would be sufficient to recover the spoken sentence from a
list.

A first experiment with the top-12 lists of one speaker (F-10;
finite-state language model) showed that among those 158 of the 200
spoken test sentences which can be correctly interpreted by the
SPICOS parser (cf. [11]), the best sentence which is accepted by the
parser is in 137 cases the spoken one, while only 123 of these senten­ces
are in top-1 position. So some extra effort in the search procedure
reduced the number of not recognized sentences by a third.

For a few spoken sentences, additional time alignments have
been carried out for each sentence on the top-10 list. The results were
as could be expected from theory: In some of the cases, the best sen­tence (position 1) forced its word boundaries to the poorer scored can­dicates, thus often increasing their scores with respect to the exact ones.
The effect of recombining the best path of a word sequence with
a better path of another word sequence described in section 2. The
exact scores generally forced a reordering of the sub-optimal top-10
list. Nevertheless, these effects are small enough to be tolerated in a
real system.

3.3 Computational and Storage Costs

The storage costs of top-N DP are proportional to $N$, the length
of a list. In our system, the storage for trace-back pointers etc. in­
creased by a factor of 2$N$; the additional factor of 2 is due to the fact
that besides the usual one transition number and one back pointer, the
cost difference to the top hypothesis and a pointer to the in-list posi­tion
have to be stored.

In order to get experimental results about the computational costs of
the algorithm, five off-line recognition tests (with finite-state gram­mar)
were performed on the 200 SPICOS sentences (754 seconds of speech)
for the same (female) speaker. The word error rate is 4.7 % ((word substitutions + deletions + insertions) / number of words), the
average number of active grid points per test frame is 2344, and (at
least)4 optimal paths are lost by the search procedure due to pruning.

The result is illustrated in figure 3 which shows that the extra com­
putational cost for recovering top-N sentences, compared to the usual
search, is significant for high $N$ (here: $N = 8$) only but negligible for

EUROspeech '89, Paris, France, September 1989
small ones. Indeed, the figures show that the measured CPU times, which depend on the load of the machine, are only a rough cue for the computational costs, which increase strictly from "STANDARD" to "TOP 8" because the computations of each test job are a subset of the following one.

Nevertheless, we can conclude that extended trace back for top N can be used for reasonable values of N (e.g., N = 5) without significant effects on recognition time. For high values of N, the list sorting becomes more and more important, such that a more sophisticated implementation is needed.

SUMMARY

Based on a generalization of Bellman's optimality principle, the one-stage dynamic-programming search algorithm can be modified in order to find the top-N sentences with respect to the Viterbi criterion. A sub-optimal variant of the top-N dynamic-programming search has been presented and tested within the SPICOS system.

Experimental tests yielded quite satisfying results in two aspects: For the finite-state grammar, the majority of the sentences not correctly recognized are at least among top-4, and the deviation from the exact solution is tolerable. For reasonable values of N (e.g., N = 5), the additional storage and computational amount is small with respect to the other SPICOS requirements. Thus, it has been demonstrated that a continuous-speech recognition system based on the one-stage dynamic-programming algorithm can be modified in order to produce a top-N sentence-hypotheses list with little effort only on the level of the book-keeping for trace-back.

ACKNOWLEDGMENT

The work described was carried out in a joint Siemens - Philips - IPO (Eindhoven) project ("SPICOS") and partially funded by the German Federal Ministry for Research and Technology (BMFT) under grant No. 413-5839-ITM 8801A. The only author is responsible for the content of this publication.

<table>
<thead>
<tr>
<th>top N</th>
<th>speaker</th>
<th>F-01</th>
<th>F-10</th>
<th>M-03</th>
<th>M-10</th>
<th>total / ( \text{in} % )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>133</td>
<td>157</td>
<td>164</td>
<td>137</td>
<td>591</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>161</td>
<td>172</td>
<td>183</td>
<td>148</td>
<td>664</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>171</td>
<td>176</td>
<td>187</td>
<td>158</td>
<td>692</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>175</td>
<td>182</td>
<td>191</td>
<td>159</td>
<td>707</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>178</td>
<td>184</td>
<td>191</td>
<td>160</td>
<td>713</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>178</td>
<td>186</td>
<td>192</td>
<td>161</td>
<td>717</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>181</td>
<td>187</td>
<td>192</td>
<td>162</td>
<td>722</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>184</td>
<td>187</td>
<td>192</td>
<td>165</td>
<td>728</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>184</td>
<td>188</td>
<td>192</td>
<td>165</td>
<td>729</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>185</td>
<td>188</td>
<td>192</td>
<td>165</td>
<td>730</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>800</td>
</tr>
</tbody>
</table>

Table 1a: Correct sentences among top-N sentence lists. N = 1 is "sentence correctly recognized", Language model: finite-state syntax (perplexity 85).

<table>
<thead>
<tr>
<th>top N</th>
<th>speaker</th>
<th>F-01</th>
<th>F-10</th>
<th>M-03</th>
<th>M-10</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.5</td>
<td>5.7</td>
<td>5.5</td>
<td>13.7</td>
<td>9.1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8.7</td>
<td>4.3</td>
<td>3.5</td>
<td>11.8</td>
<td>7.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7.9</td>
<td>4.0</td>
<td>3.1</td>
<td>11.0</td>
<td>6.5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7.5</td>
<td>3.7</td>
<td>2.7</td>
<td>10.6</td>
<td>6.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7.3</td>
<td>3.5</td>
<td>2.6</td>
<td>10.2</td>
<td>5.9</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7.0</td>
<td>3.2</td>
<td>2.6</td>
<td>10.0</td>
<td>5.7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6.8</td>
<td>3.0</td>
<td>2.5</td>
<td>9.4</td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6.5</td>
<td>2.7</td>
<td>2.4</td>
<td>9.1</td>
<td>5.2</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>6.3</td>
<td>2.7</td>
<td>2.2</td>
<td>9.0</td>
<td>5.1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>6.1</td>
<td>2.6</td>
<td>2.1</td>
<td>8.8</td>
<td>4.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Top-N error rates (defined in section 3.2) in % (bigram language model used).

REFERENCES