Enhancements of Viterbi Search for Fast Unit Selection Synthesis

Daniel Tihelka, Jiří Kala, Jindřich Matoušek

Dept. of Cybernetics, Faculty of Applied Sciences, University of West Bohemia, Czech Rep.
dtihelka@kky.zcu.cz, jkala@kky.zcu.cz, jmatouse@kky.zcu.cz

Abstract

The paper describes the optimisation of Viterbi search used in unit selection TTS, since with a large speech corpus necessary to achieve a high level of naturalness, the performance still suffers. To improve the search speed, the combination of sophisticated stopping schemes and pruning thresholds is employed into the baseline search. The optimised search is, moreover, extremely flexible in configuration, requiring only three intuitively comprehensible coefficients to be set. This provides the means for tuning the search depending on device resources, while it allows reaching significant performance increase. To illustrate it, several configuration scenarios, with speed–up ranging from 6 to 58 times, are presented. Their impact on speech quality is verified by CCR listening test, taking into account only the phrases with the highest number of differences when compared to the baseline search.

Index Terms: speech synthesis, unit selection, Viterbi search, space search pruning

1. Introduction

The unit selection technique is known for its ability to produce nearly natural-sounding synthetic speech, but also for its huge hardware requirements necessary to achieve the quality. Speech corpora containing tens of hours of speech are not rare in this technique. While storage and memory do not represent such a big limitation, finding the best candidate sequence in the graph of candidate instances may require a not-negligible amount of time. This is especially crucial for server applications which must dispatch several simultaneous synthesis requests as fast as possible.

In the search process, it is the complexity of the join cost[1] computing which consumes the largest part of the computation time, as the cost must be evaluated in run-time, one order of magnitude more often than the target cost[2]. In the past, various techniques attempting to bypass the join cost computing were presented. For example, the most frequent subset of join cost values can be cached to avoid their computing [1]. This does not affect the selected sequence, as all candidates are still considered in the selection process, but due to cache search complexity and its size, it is suitable rather for smaller corpora. Alternative approaches reduce the number of computations by the reduction of unit candidates number, employing miscellaneous preselection techniques based on various statistics [2, 3]. This, however, may change the selected sequence, as some candidates are excluded from the selection.

This work was supported by the Grant Agency of the Czech Republic, project No. GACR 102/09/0989, and by the Ministry of Education of the Czech Republic, project No. 2C06020. The access to the MetaCentrum computing facilities was supported by the research intent MSM6383917201.

[1] Evaluates unit candidates join smoothness; also known as concatenation cost.

[2] Evaluates prosodic properties of units related to what is required.

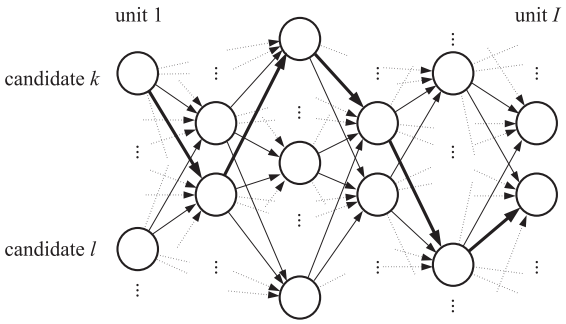

Figure 1: *The schematic diagram of units and theirs candidates searched in unit selection by Viterbi algorithm.*

An original approach has been chosen in [4], where the Viterbi search (used in a majority of unit selection systems, including those referenced) was enhanced by two stopping criteria. They, in general, do not allow the computation of join cost in cases where no better cumulative cost[3] can be found for a unit. Our present paper is based on this approach, which is further extended. Contrary to [4], where only algorithm speed was evaluated, we have also carried out listening tests to evaluate the impact of various algorithm settings on the quality of generated speech.

2. Viterbi search algorithm optimisation

Having the sequence of I units to synthesize, let in the paper

$$TC_i(k) \geq 0, \quad k = 1, \ldots, K_i, \quad i = 1, \ldots, I$$

represents the value of target cost computed for the k-th candidate of i-th unit (matched against the target specification of the i-th unit),

$$JC_{i-1,i}(k,l) \geq 0, \quad k = 1, \ldots, K_{i-1}, \quad l = 1, \ldots, K_i, \quad i = 2, \ldots, I$$

is the value of join cost computed between k-th candidate of unit $i-1$ and l-th candidate of unit i (all $K_{i-1}K_i$ combinations of candidates are computed and examined), and

$$C_i^*(k), \quad K_i^*(k), \quad k = 1, \ldots, K_i, \quad i = 1, \ldots, I$$

contains the best cumulative cost for the k-th candidate of i-th unit, and the index of its best predecessor, computed as

$$C_i^*(k) = \begin{cases} TC_1(k), & i = 1 \\ \min\limits_l \Big(C_{i-1}^*(l) + JC_{i-1,i}(l,k) + TC_i(k) \Big), & i > 1 \end{cases}$$

$$K_i^*(k) = \begin{cases} \text{undefined}, & i = 1 \\ \arg\min\limits_l \Big(C_{i-1}^*(l) + JC_{i-1,i}(l,k) + TC_i(k) \Big), & i > 1 \end{cases}$$

[3] The sum of TC and JC for a sequence of candidates.

2.1. Baseline search algorithm

For the sequence of $i = 1, \ldots, I$ units and their target specifications, the scheme of the basic Viterbi is:

1. Initialise for the first unit:

   `foreach` $k = 1, \ldots, K_1$
   `compute` $TC_1(k)$
   `set` $C_1^*(k)$

2. Compute for all other units $i = 2, \ldots, I$:

   `foreach` $k = 1, \ldots, K_i$
   `foreach` $l = 1, \ldots, K_{i-1}$
   `compute` $JC_{i-1,i}(k, l)$

   `compute` $TC_i(k)$
   `set` $C_i^*(k)$ and $K_i^*(k)$

3. Find the sequence of the best candidates:

   `set` $k_I^* = \arg\min_k C_I^*(k)$
   `backtrack` $k_i^* = K_{i+1}^*(k_{i+1}^*)$

   where k_i^* is the index of the i-th unit candidate used to create the synthetic speech.

The basic scheme can be substantially (see Table 1) improved by a simple heuristics – when the cumulative cost of l-th predecessor of the current candidate k is larger than the lowest cumulative cost of the candidate k obtained so far, the join cost to the l-th candidate does not have to be evaluated, since the candidate will never by chosen as the best predecessor $K_i^*(k)$ (whatever the join cost value, see $C_i^*(k)$ computing). The modified search then works as:

1. Initialise for the first unit equally to the basic Viterbi algorithm.
2. Compute for all other units $i = 2, \ldots, I$:

   `foreach` $k = 1, \ldots, K_i$
   `foreach` $l = 1, \ldots, K_{i-1}$
   `if` $C_{i-1}^*(l) > \min_{j=1,\ldots,l-1} \left(C_{i-1}^*(j) + JC_{i-1,i}(k, j) \right)$
   `set` $JC_{i-1,i}(k, l) = \infty$
   `else`
   `compute` $JC_{i-1,i}(k, l)$

   `compute` $TC_i(k)$
   `set` $C_i^*(k)$ and $K_i^*(k)$

3. Find the sequence of the best candidates equally to the basic Viterbi.

This simple condition speeds up the search approximately by factor 4, and therefore, it has been used in ARTIC TTS [5] since embedding the unit selection approach. Thus, in the paper it will be referred to as *baseline algorithm*.

2.2. Enhanced search algorithm

In [4], the Viterbi search was modified with the aim to avoid examining unnecessary joins, while the correct natural expectation may be to neglect the paths with very bad cumulative cost. First of all, the authors defined *beam width* pruning constant K_Θ, which limits the number of candidates for each unit i to the (globally) defined number. Moreover, two *admissible stopping* criteria were embedded into the algorithm, which, similarly to the heuristic stopping in the baseline algorithm, stop the search in cases where no better cumulative cost can be found. Both criteria expect the C_i^* being sorted in ascending order[4], and work as follows:

Admissible stopping of candidates evaluation (called *admissible stopping for the beam* by the authors) can stop the examining of candidates for each unit i when $K_i > K_\Theta$, and when a candidate $k > K_\Theta$ cannot reach lower cumulative cost than the highest cumulative cost reached among candidates $k = 1, \ldots, K_\Theta$. If such candidates are to be evaluated, they would appear under the K_Θ threshold anyway, after the cumulative costs sorting — see Figure 2.

To determine the candidates not to be examined, the value of minimum possible join cost among all candidate combinations for a particular unit join

$$JC_{i-1,i}^{\min} = \min_{k,l} \left(JC_{i-1,i}(k, l) \right) \quad k = 1, \ldots, K_{i-1}, \quad l = 1, \ldots, K_i$$

must be known. It can be pre-computed in advance for all meaningful unit joins (e.g. for diphones *[ab–bc]*, but not for *[ab–cd]*), and its value will be 0 for unit joins having at least one candidate combination neighbouring in the corpus. Accordingly, if the value is not pre-computed, it can simply be set to 0

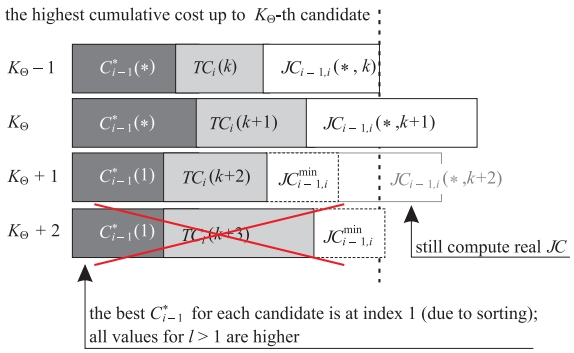

Figure 2: *The scheme of admissible stopping of candidates evaluation. Stop search if for given $TC_i(k)$ we cannot get a better cost than the dashed line. The symbol $*$ represents the index of the best predecessor of the candidate.*

Admissible stopping of JC computing (called *admissible stopping in local minimisation* by the authors) can stop the join cost computing, *with no approximation error*, when no predecessor $l > 1$ of a candidate k can reach lower cumulative cost than the lowest $C_i^*(k)$ reached for $j = 1, \ldots, l-1$ so far — see Figure 3.

The idea is the same as the heuristics in the baseline algorithm, although it is more sophisticated in benefiting from C_{i-1}^* sorted order and the use of $JC_{i-1,i}^{\min}$.

The search algorithm with both search stopping criteria, as presented in [4], now looks as follows:

1. Initialise for the first unit equally to the basic Viterbi algorithm, plus do the following

   ** *Keep only K_Θ best candidates* **
   `sort` $C_1^*(k)$
   `set` $K_1 = K_\Theta$

[4]Note that the authors use term *score* in [4], instead of more common *cost* we use in this paper. Due to the reversed relation of the terms, we inverted the descriptions to be valid for the use *cost*.

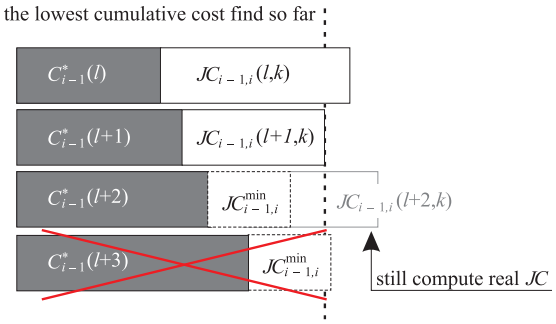

Figure 3: *The scheme of admissible stopping of JC computing. Stop search if we cannot get a better cost than the dashed line, while $TC_i(k)$ is constant $\forall l$ here.*

2. Compute for all other units $i = 2, \ldots, I$:

```
foreach  k = 1, ..., K_i
  compute  TC_i(k)

  ** Admissible stopping of candidates evaluation **
  if  k > K_Θ  and
      C*_{i-1}(1) + TC_i(k) + JC^min_{i-1,i} > max_{j=1,...,K_Θ} ( C*_i(j) )
    set   K_i = k
    next  k

  foreach  l = 1, ..., K_{i-1}
    ** Admissible stopping of JC computing **
    if  C*_{i-1}(l) + JC^min_{i-1,i} > min_{j=1,...,l-1} ( C*_{i-1}(j) + JC_{i-1,i}(k,j) )
      break
    else
      compute  JC_{i-1,i}(k,l)

  set  C*_i(k) and K*_i(k)

** Keep only K_Θ best candidates **
sort  C*_i(k)
set   K_i = K_Θ
```

3. Find the sequence of the best candidates equally to the basic Viterbi.

This optimisation itself is, even with $K_\Theta = \infty$ (i.e. no pruning), approximately 1.5-times faster than the baseline algorithm. However, for lower resource devices or highly loaded servers, it is still not sufficient.

2.3. Pre-pruning after target cost evaluation

By the analysis of the optimised selection behaviour, using the corpus and phrase set described in Section 3, we have found that even for $JC^{\min}$ set, the majority of candidates are still searched for $k > K_\Theta$ in admissible stopping of candidates evaluation before the loop is left. More specifically, it is 99.8% for $K_\Theta = 400$ and 99.7% for $K_\Theta = 200$, where 100% is the total number of candidates searched to synthesise all the phrases: $\sum_{\text{phrases}} \sum_i K_i$. The reason is simple — the corpus size and its rich diphone coverage.

To further enhance the selection speed, we, therefore, have to employ one additional pruning constraining the maximum number of candidates examined in each step i well before K_Θ search stopping is applied. This pre-pruning is controlled by constant K_T increased by $K_\%$ percent of unit candidates, i.e. $K_\%(n) = n * (K_\%/100)$. For meaningful pruning configurations, it is feasible to ensure:

$$K_\Theta < K_T + \max_i \Big(K_\%(K_i) \Big)$$

while setting $K_T = \infty$ switches the pre-pruning entirely off. In this way, the upper limit of the candidates number searched after K_Θ threshold can be defined while still keeping the candidates cardinality for each unit into account.

The pre-pruning is thus embedded into the optimised Viterbi search from the previous section as follows:

1. Initialise equally to the optimised algorithm from Section 2.2.
2. Compute for all other units $i = 2, \ldots, I$:

```
foreach  k = 1, ..., K_i
  compute  TC_i(k)

** Keep only K_T + K_%(K_i) best candidates **
sort  TC_i(k)
set   K_i = min(K_T + K_%(K_i), K_i)
```

3. Continue by step 2 in the optimised algorithm from Section 2.2, using the already computed TC_i values.
4. Find the sequence of the best candidates equally to the basic Viterbi.

3. Selection speed–up evaluation

All the experiments described further were carried out with our speech corpus consisting of 12,277 recorded sentences with 17 hours and 49 minutes in duration excluding pauses [6]. As the constants $K_T, K_\%$ and K_Θ provide extreme flexibility in configuration, we need some cue points from which the impact of search configuration on the speed can be extrapolated. Therefore, we analysed various constant combinations uniformly spread through the configuration space[5], from which we have selected 4 configuration scenarios for further analysis:

conservative	K_T =600,	$K_\%$ =10,	K_Θ =500
optimal	400,	10,	400
aggressive	200,	10,	100
extreme	100,	10,	50

To measure the speed of the basic, baseline, and the optimised search, we have selected 40 phrases at random and logged the number of join cost computations for each individual search – the measure of speed–up by the comparison of join cost computation numbers excludes the dependency on actual hardware load (and its changes), and allows running several experiments in parallel. To synthesise the 40 phrases, 1821 candidates (1772 unique) were joined, while the number of candidates for each unit in the sequence was 1027 on average. This is a significantly larger test set, as well as significantly larger corpus, by means of which the results are collected, than that presented in [4]; the results of all the algorithms are summarised in Table 1.

Table 1: *The comparison of speed–up for all the examined search versions, measured on the number of join cost computations. For the basic version of search, the JC count was* 6,042,290,874 *and TC count was* 3,286,727.

$K_T, K_\%, K_\Theta$	*JC* count	To baseline	To basic
baseline	1,443,475,302	–	0.24
∞, 0, ∞	947,574,039	1.5	6.4
600, 10, 500	249,342,268	5.8	24.2
400, 10, 400	172,804,926	8.4	35.0
200, 10, 100	54,478,568	26.5	111.0
100, 10, 50	24,717,692	58.4	244.5

[5]The details are not very interesting as regards this paper, as basically any constant combinations with reasonable values spread could be chosen, displaying the same tendencies.

Let us note that all numbers were collected with JC^{min} precomputed. However, it does not have any significant impact on the results, as presented at the beginning of Section 2.3.

4. The impact on synthetic speech quality

The evaluation through listening tests is necessary for a reliable conclusion about the impact of various pruning configurations on the synthetic speech, as it can be assumed that the fewer units remain in the selection process, the lower the speech quality will be. The important issue is the choice of sentences for the test, as carrying out the test on a small number of randomly selected sentences cannot in general ensure the validity of results.

Therefore, for the baseline algorithm and for all the pruning configurations described in Section 3, we have synthesised more than $500,000$ sentences, containing $965,905$ unique phrases, and the unit candidates chosen for each synthesised phrase have been logged. Then we have matched the phrases from the optimised Viterbi search to its baseline version, and excluded all the phrases consisting of the equal candidate sequences (each candidate was unambiguously identified by its name and position in the speech corpus). The number of remaining phrases is shown in Table 2.

Selected sequences in the differing phrases have been then compared candidate-by-candidate, and scored according to the number of different candidates (relative to the number of candidates in the phrase) and the ratio of join points (where candidates not neighbouring in the corpus are concatenated). Then, for each pruning configuration independently, $10 + 10$ phrases, ranging from 5 to 10 words, with the worst score have been chosen, resulting in 80 unique phrase-pairs for the listening test.

Such an approach is fairly unique, as it gives us the exact number of cases where both approaches compared produce equal speech which does not have to be tested then. Moreover, regarding the differing variants, it focuses on the worst cases (those most differing from the baseline), giving us the most pessimistic estimate of the evaluated system behaviour. Let us also note that according to the restriction of listening test scale, when the number of differing sentences is increasing, the reliability of such a selection approaches in limit[6] random sentence selection which is still the best one in the case of no prior knowledge of the data.

In the test, we had 20 participants without any known hearing problems, who have compared all the 80 phrase-pairs, shuffled by random with random AB/BA ordering, and evaluated them on 5-point Comparison Category Rating scale:

-2 A variant sounds much better
-1 A variant sounds better
0 no preference between A and B
1 B variant sounds better
2 B variant sounds much better.

The results were then normalised to A be the baseline system and B be one of the optimised variants.

It can be seen from Table 2 that considering the comparison of the most differing variants of synthetic speech, the search optimisations does not have any negative impact on its quality. Surprisingly, it is true even for the the most "aggressive" configuration (more than $58\times$ faster), although one may object that due to the large number of differing phrases in that case, we cannot be sure that some other phrases would not show serious quality deterioration. Still, such a choice is better than random selection, and standard deviation and Figure 4 confirm that listeners tend to choose the answer "no preference" most of the time, and its nearest neighbours roughly equally, whatever the search configuration evaluated.

[6] The limit is here the case when all the sentences differ equally.

Table 2: *The comparison of the selected search configurations to the baseline. The "ϕ score" column shows the average listener's CCR score and its standard deviation σ, the "diff. phrases" shows the number and percentage of the phrases differing from the baseline in at least one unit.*

K_T, $K_{\%}$, K_{Θ}	ϕ score	$+/-\sigma$	diff. phrases
600, 10, 500	-0.036	$+/-0.831$	46,186 (4,8%)
400, 10, 400	-0.044	$+/-0.886$	75,958 (7.8%)
200, 10, 100	0.099	$+/-0.842$	478,434 (49.5%)
100, 10, 50	-0.068	$+/-0.810$	692,417 (71.7%)

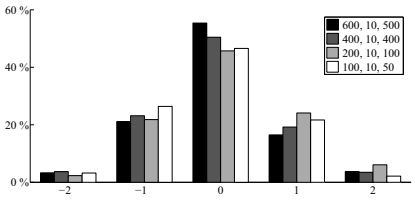

Figure 4: *The histogram of listening test scoring distribution.*

5. Conclusion

The proposed Viterbi search enhancements can significantly speed–up the the search in unit selection TTS system, with hardly noticeable impact on the quality of synthetic speech. This has been confirmed by the larger–scale listening test, focusing on the cases when the enhanced selection differs the most significantly from the baseline variant, which gives us the most pessimistic estimate of the quality.

Moreover, the search is extremely flexible in configuration, while there are only three intuitively comprehensible coefficients, which can be easily explained even to speech synthesis non-experts.

6. References

[1] Beutnagel, M., Mohri, M., Riley, M., "Rapid unit selection from a large speech corpus for concatenative speech synthesis", in *Proceedings of EUROSPEECH'99*, vol. 2, Budapest, Hungary, 1999, pp. 607–610.

[2] Hamza, W., Donovan, R., "Data-driven segment preselection in the IBM trainable speech synthesis system", in *Proceedings of ICSLP 2002*, Denver, Colorado, USA, 2002, pp. 2609–2612.

[3] Matoušek, J., Tihelka, D. and Hanzlíček, Z.: "Reducing footprint of unit selection TTS system by excluding utterances from source speech corpus", in *Proceedings of 19th Czech-German Workshop on Speech Processing*, Prague, Czech Republic, 2009, pp. 92–98.

[4] Sakai, S. Kawahara, T. Nakamura, S., "Admissible stopping in viterbi beam search for unit selection in concatenative speech synthesis", in *Proceedings of IEEE-ICASSP*, Las Vegas, US, 2008, pp. 4613–4616

[5] Matoušek, J., Tihelka, D. Romportl, J.: "Current state of Czech text-to-speech system ARTIC", in *Text, Speech and Dialogue*, ser. Lecture Notes in Artificial Intelligence. Berlin, Heidelberg: Springer, 2006, vol. 4188, pp. 439–446.

[6] Matoušek, J., Tihelka, D. Romportl, J.: "Building of a speech corpus optimised for unit selection TTS synthesis", in *Proceedings of LREC 2008*, Marrakech, Morocco, 2008.