Unit Selection Cost Function Exploration Using an A* based Text-to-Speech System

David Guennec and Damien Lolive

IRISA - University of Rennes 1, Lannion, France
{david.guennec,damien.lolive}@irisa.fr

Abstract. Speech synthesis systems usually use the Viterbi algorithm as a basis for unit selection, while it is not the only possible choice. In this paper, we study a speech synthesis system relying on the A^* algorithm, which is a general pathfinding strategy developing a graph rather than a lattice. Using state of the art techniques, we propose and analyze different selection strategies and evaluate them using a subjective evaluation on the N-best paths returned. The best strategy achieves a MOS score of 3.29 ($\pm$0.18). More interesting, the proposed system enables an in-depth analysis of unit selection.

Keywords: Corpus-based Unit Selection TTS, Evaluation, Cost function accuracy, Concatenation cost, Cost weighting.

1 Introduction

Currently, two main techniques dominate the field of Text-to-Speech synthesis. One of them is the parametric approach for which HTS [1] is the main system. The second one is the historical concatenative approach extended by [2] and widely implemented, for example in [3,4,5,6,7]. Even if the majority of research works concern the HTS framework, it is the corpus-based approach which is mainly used in commercial systems. In both cases, the current main research problem is expressivity control during synthesis [8]. Then, to enable the exploration of expressive corpus-based synthesis, we have to explore the in-depth behavior of the system and especially cost functions.

In corpus-based synthesis, the fundamental question is how the most appropriate sequence of units in an annotated speech corpus can be efficiently selected? This most appropriate units sequence is generally found by minimizing two criteria: the number of concatenations and the risk of generating hearable concatenation artefacts. Numerous criteria have been proposed in [9] but generally the cost function to optimize follows the form [3]:

$$U^* = \underset{U}{argmin} \left(\sum_{n=1}^{card(U)} C_t(u_n) + \sum_{n=2}^{card(U)} C_c(u_{n-1}, u_n) \right) \quad (1)$$

where U^* is the best unit sequence according to the cost function and u_n the candidate unit in the candidate sequence U intending to match the n^{th} target unit. The sub-cost $C_t(u_n)$ (target cost) represents the distance between candidate and corresponding target unit. The second sub-cost (concatenation cost), $C_c(u_{n-1}, u_n)$, is the distance between

P. Sojka et al. (Eds.): TSD 2014, LNAI 8655, pp. 432–440, 2014.

the current candidate u_n and the previous one u_{n-1} in the path, used to minimize artifacts in concatenation areas.

Since unit selection has to process millions of candidate units, pruning is needed to give a result in an acceptable time or even in real time for most industrial applications. In most cases, the Viterbi algorithm [10] is used to find the best unit sequence (for example in [4,11,7]), but it is not the only possible one. An attempt to use a genetic algorithm in [12] can also be mentioned. Practically, the Viterbi algorithm is optimal and achieves a complexity of $\mathcal{O}(N * K^2)$, K being the maximum of candidate units for a given target phoneme. Since unit selection can be formulated as a path finding problem, other graph exploration algorithms can also be applied. In particular, some algorithms like A^* are better-suited for heuristic introduction to speed up the unit selection step.

In this paper, we describe in Section 2 a new unit selection based TTS system relying on the A^* algorithm, built to analyze cost functions behavior and exploration strategies. Then the general architecture of the system is proposed in Section 3. A description of the corpus used follows in Section 4 and the experimental setup is depicted in Section 5. Finally, the results are presented in Section 6. The paper is viewed as a preliminary study aiming at demonstrating the feasibility of using a A* algorithm to drive a unit selection process. The second objective of the paper is to establish a reference set of preselection filters and cost function for a comparison with other algorithms.

2 Unit Selection as a Pathfinding Problem

In this section we first introduce the A^* algorithm used to find the best unit sequence according to the cost function. Then, some implementation details are given.

2.1 *A** Algorithm

Contrary to the Viterbi algorithm, which computes a lattice containing all the candidate nodes (or at least M nodes for each time instant), A^* algorithm develops a graph. At each time instant, it explores the best node of the graph using a cost function that depends on both the path from the source node and the estimated cost to the target.

Originally introduced in 1968 by [13], the algorithm basically operates by searching for a path in a directed graph, whose nodes only have a finite number of successors, between a start node and a target node. To avoid arbitrary choice of the start node, we introduce a dedicated start node s which has the first candidate units as successors. We make a similar choice by introducing a unique target node t. 614bibThe algorithm uses a cost function of the form $f(n) = g(n) + h(n)$ with $g(n)$ being the cost of the sub-path between s and current node n and $h(n)$ being the estimated (heuristic) cost between n and t.

At each step, A^* takes the most promising node according to $f(n)$ and expends its successors (computing $f(n)$ by the way) until t is reached. $h(n)$ is a heuristic that enables to speed up the algorithm by privileging the nodes that seem to be on an optimal path over those which have a better $g(n)$ cost but may lead to greater costs in the future [14].

Considering a unique target node, one of the main advantages of A^* is that the algorithm delivers an optimal solution if the heuristic is admissible, ie. if $h(n) \leq h^*(n)$, where $h^*(n)$ is the real minimum value of the distance to the target node. In particular, note that the algorithm is optimal in the trivial case $h(n) = 0$, ie. if there is no heuristic, and turns out to be equivalent to Dijkstra's algorithm.

2.2 Adaptation to the unit selection problem

Considering the algorithm presented above, the main functions that need to be adapted to our problem are (1) the cost function computation and (2) the successor function. In this work, we only consider the $g(n)$ part of the cost function, thus putting $h(n)$ to 0 which insures algorithm optimality. The cost function is detailed in Section 3.2.

Concerning the successor function, it needs to consider domain-based knowledge. During the search process, each phone of the target sequence is considered as the start of a potential unit for developing the graph. Moreover, for a particular unit, all the candidates within a window of width $\omega = 5$ are explored. Window width has been chosen arbitrarily to give satisfying search speed.

Furthermore, to improve algorithmic performance, the OPEN list is implemented as a binary heap sorted according to the cost function and a joined hash table to get quick membership queries. In addition, all the graph nodes are not computed, only those expanded during the successors search are really created.

In order to explore cost functions behavior, we modified the algorithm to be able to get the N-best paths, and also to get the N-best paths between a minimum and a maximum cost.

3 System Architecture

The system is interfaced with the Roots toolkit as described in [15]. For the synthesis, feature extraction starts from Roots files (generated from textual source using automatic tools) containing the target sequence with the needed annotations. Then the unit selection step is done using the A^* algorithm presented in Section 2. This step is parameterized by a cost function and user parameters (for example, requesting the best path or the N-best paths). The selection process makes queries to the corpus through the pre-selection filters as explained in Section 3.1. Finally, signal generation is performed by mixing each two units on an horizon of 2 pitch periods, using Hann windows.

3.1 Pre-selection filters

As the problem of searching for variable-sized units in a corpus is computationally expensive, hash tables and pre-selection filters are implemented to speed up the unit selection process [11]. To achieve this, a key containing discrete information (mostly binary) is created for each speech segment (phoneme or non speech sound) in the corpus. That enables A^* to take or reject the unit quickly by just comparing the values in the key with target values. The key contains phonetic, linguistic and prosodic information. Binary masks are used to get access only to the desired information during runtime.

The pre-selection filters, using data included in segments keys, are integrated to the hash functions used to access the units in the corpus in order to reduce the number of candidates added to the OPEN list. The set of filters used for these experiments is the following, for each speech segment constituting the unit:

1. Is the segment a non speech sound?
2. Is it in the onset of the syllable?
3. Is it in the coda of the syllable?
4. Is it in the last syllable of its breath group?
5. Is the current syllable in word end?
6. Is the current syllable in word beginning?

In the case of a non speech sound, the only feature that matters is the first one, the others being all set to false. if no unit corresponding to the current set of filters is found, the pre-selection filters are relaxed one by one, starting from the end of the list. The priority order of the filters is the one given above. One drawback is that we can explore candidates far from the target features we want, thus risking to produce artefacts but this backtracking mechanism insures to actually find a unit and so insures the production of a solution.

3.2 The target and concatenation costs

The cost of a unit is generally divided into a target cost and a concatenation cost as described in equation (1). Here, only the concatenation cost is included in the function, the target cost part being achieved by the filters. Weights are introduced here to balance the magnitude differences of the sub-costs.

Concatenation cost Here, we consider three sub-costs, well rated in the state of the art, for concatenation quality evaluation which are MFCC, amplitude and f0 distances, all weighted to give equal importance to each sub-cost on a phoneme basis. Basic rules addressing duration where first included and then dropped, first because they did not show a real improvement and also because generated speech seems generally well enough. Nevertheless, the inclusion of real duration or intonation models would be very interesting. Hence, we have the following concatenation cost:

$$C_c(u,v) = W_{mfcc}(u,v)C_{mfcc}(u,v) + W_{amp}(u,v)C_{amp}(u,v) + W_{F0}(u,v)C_{F0}(u,v)$$

where u and v are the two units under comparison, $W_{mfcc}(u,v)$ is the normalization coefficient for the MFCC cost $C_{mfcc}(u,v)$, $W_{amp}(u,v)$ is the normalization coefficient for the amplitude cost $C_{amp}(u,v)$ and $W_{F0}(u,v)$ is the normalization coefficient for the F0 cost $C_{F0}(u,v)$.

Weights tuning is a completely distinct problem, and many methods have been experimented. [16] includes a good review of the most common techniques. Actually, subjective methods remain widely used among TTS systems, in the absence of objective methods strongly correlated to perception. Here, we consider that the weights are

linked to the phoneme to be concatenated and are computed using means and standard deviations for each phoneme of the corpus.

In addition, given that in particular context-dependent and voiced phonemes are more susceptible to produce concatenation artefacts than others because of higher spectrum, pitch and energy variability, we decided to evaluate the relevance of a feature addressing this particular issue. Different additions to cost functions have been experimented [9] [17], but the one we viewed as the most promising was only used, to our knowledge, for corpus reduction. This feature, "vocalic sandwich" cost [18] [19] is a sequence of phonemes following the regular expression $C(W^*VW^*)^+C$ where V is the vowels set, C a set including all the consonants considered to feature good splicing properties and W contains the remaining. In our implementation, we put a penalty to the units not respecting the sandwich expression.

Target cost In our implementation, the target cost is not directly incorporated in the cost function. Indeed, we consider that there is no need to integrate the nodes failing to show a certain fitting to the target sequence in the graph. As other works showed, the nodes achieving a good target cost are generally equally satisfying. Hence, features used for preselection also stand as binary target sub-costs. This means there is no target cost mark in our implementation, units are processed by pass or reject preselection filters. As the values we use are binary, their integration into the preselection filters is easy. Thus, the units that satisfy a given level of filters are considered equivalent regarding the target cost.

4 Corpus Analysis

For all the experiments, we used a French expressive corpus built from an audiobook spoken by a single male speaker. The corpus is automatically annotated using the global process described in [15] and is represented using the Roots toolkit. Speech signal is sampled at 44.1kHz (1 channel) and the mean F0 value for the speaker considering voiced segments is as low as 87Hz.

Since it is an audiobook, the content of the corpus and expressivity are not controlled, which may be a problem. For instance, as the voice has not been recorded for TTS purposes, prosody is sometimes exaggerated. Nevertheless, such a corpus has some interesting properties like homogeneous speech or good signal quality.

Concerning the size of the full corpus, it consists of 3 339 utterances, totalizing 419 742 speech segments (388 251 phonemes & 31 491 non speech sounds). The overall length of the corpus is 10h 45'. Diphoneme covering (and other non speech sounds) is 78%. The missing diphones are most of the time unused in French or even impossible.

The full corpus is then split into three sub-corpora: learning (3 139 utterances), testing (100 utterances) and validation (100 utterances). Testing and validation corpora are manual checked and represent about 25' each.

5 Experiments

5.1 Experimental setup

The experiments we have conducted intend to:

1. Prove that a TTS system using A* to drive a unit selection process is viable.
2. Demonstrate the ability of the system to extract the N-best paths easily.
3. Assess the overall performance of the system.
4. Establish a reference cost function for further experiments.
5. Verify the stability of the cost functions presented above: the main idea is to explore the variability & ranking accuracy for the 100 best paths found by our system.

Figure 1 shows the evolution of the global cost for the 100 best paths found. The target sentence is ”Car ce n’est pas le chagrin qui la fit partir” (Because it is not grief that caused her to leave). Due to the differences between selected units among the paths, costs are reported to each phoneme on the sequence (x axis). At first glance, we observe little variability among the paths. Most changes seem to occur on the first/last units (Non Speech Sounds here actually). Due to lack of space we cannot show figures for individual sub-cost illustrating the compensations existing between them. The mean number of units passing the filters is approximately 200, with a mean size of 3.8 phones for selected units, which is satisfying. We noted that the relaxation of filters was quite rare, which could signify other filters can be added to refine our target cost.

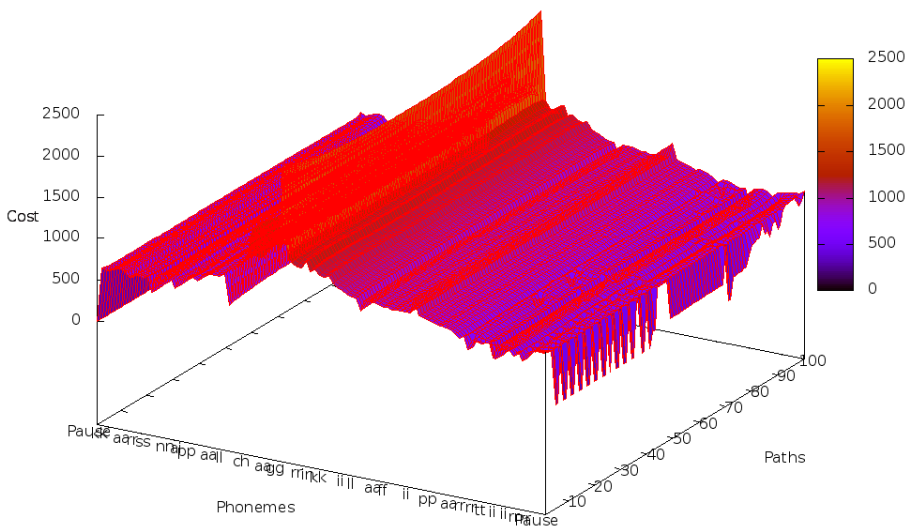

Fig. 1. Global cost evolution for 100-best paths (French sentence ”Car ce n’est pas le chagrin qui la fit partir”).

Three listening tests were also accomplished. For each, we have followed ITU-T P.800 recommendation, with 8 expert testers, questions/answers being those described in the recommendation.

6 Results and Discussion

6.1 Overall quality

The first subjective test is a MOS test intending to rate the global performance of two variants of the system: (1) *Base* with all weights set to 1, (2) *Smoothing*: adjusted weights but no sandwich, (3) *Sandwich*: weights set to 1 + sandwich cost and finally (4) *All*: adjusted weights + sandwich costs. Each listener evaluates 20 samples for each system.

The results of the MOS test are shown on Figure 2 on the left. The *Smoothing* function gives the best overall quality with a MOS score of 3.29 (± 0.18). This result is better than *Base*'s result (2.93 ± 0.17). Surprisingly, *Base* achieves a reasonably good performance considering the distribution of the sub-costs. Indeed, an unweighted amplitude cost tends to upper bound the values of other sub-costs. Then, we can think that the amplitude cost has a significant impact (in well) on listeners' decisions. More analysis on the individual sub-costs, particularly correlation tests, should be performed to complete this result. On the contrary, the use of sandwich penalties seems inconclusive (3.05 (± 0.18)), even when combined with weighted sub-costs(2.98 (± 0.20)). We believe that trying to find a vocalic sandwich path may result in worse fitted units on the prosodic plan than those selected only over spectral considerations.

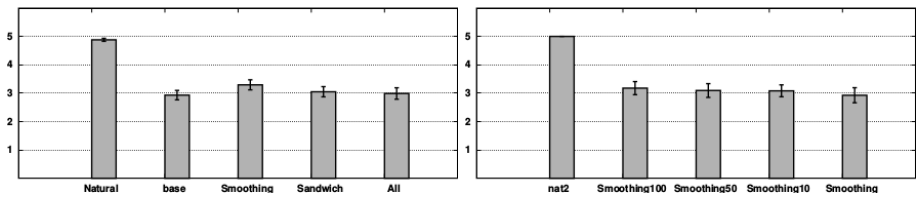

Fig. 2. MOS test between the 4 variants of the cost function (left) and DMOS results for the 1^{st}, 10^{th}, 50^{th} and 100^{th} paths of the *Smoothing* cost function (right).

6.2 Behavior of the cost function with the N-best paths

Given these results, we have decided to conduct a DMOS test involving the *Smoothing* system which is the best system according to preceding tests.

Each time, the natural signal is confronted to a synthesized signal corresponding to the 1^{st}, 10^{th}, 50^{th} or 100^{th} path according to unit selection and a duplicated natural reference. Each listener hears the 12 same sentences for each system. Figure 2 (right part) shows results of the test. The results for all paths are approximately 3, meaning slightly annoying degradation. Confidence intervals are thiner but no preference can

be observed. No correlation between functions rankings and the testers choices are observed and minimal variability is spotted. These results raise many interrogations. First, the concatenation costs or the distance used may not be accurate. Secondly, as other works also pointed out low variability over the first paths, it would be more relevant to lookup further, up to the 1 000 first paths for example. These questions should be considered in order to enhance the ranking function.

7 Conclusion

In this paper, we have presented a new speech synthesis system with an original unit selection algorithm implemented as an A^* rather than the usual Viterbi. The system, designed as an experimental platform to explore the behavior of concatenative speech synthesis in depth, implements state of the art mechanisms to perform the unit selection. In particular, using A^* makes it very easy to explore the N-best paths in a unit selection problem while a good heuristic can drastically improve the time necessary to sort the problem out without sacrificing the result's optimality. Several cost functions have been evaluated for this new system, showing that vocalic sandwiches, very efficient for corpus reduction, do not enhance speech quality when used as part of the selection process. Furthermore, a reference cost function, *Smoothing*, has been established for further experiments. Further works will focus on 3 aspects: (1) Filters refinement, (2) cost function improvement and finally (3) comparing our A* with Viterbi considering efficiency, speed and overall quality using the same set of filters, cost function and corpora.

References

1. J. Yamagishi Z. Ling and S. King, "Robustness of HMM-based Speech Synthesis". In Proc. of the international conference on speech prosody, pp. 581-584, 2008.
2. Y. Sagisaka, "Speech synthesis by rule using an optimal selection of non-uniform synthesis units". In Proc. of IEEE ICASSP, pp.679-682, 1988.
3. A. W. Black and P. Taylor, "CHATR: a generic speech synthesis system". In Proc. of the 15^{th} conference on Computational linguistics, Volume 2, pp. 983-986, 1994.
4. A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database". In Proc. of IEEE ICASSP, Vol. 1, pp. 373–376, 1996.
5. P. Taylor, A. Black, and R. Caley, "The architecture of the Festival speech synthesis system". In Proc. of the ESCA Workshop in Speech Synthesis, pp. 147–151, 1998.
6. A. P. Breen and P. Jackson, "Non-uniform unit selection and the similarity metric within BT's Laureate TTS system". In Proc. of the ESCA Workshop on Speech Synthesis, pp. 373–376, 1998.
7. R. A. . Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the Festival speech synthesis system", Speech Communication, vol. 49, no. 4, pp. 317–330, 2007.
8. A.R.F. Rebordao, M. Shaikh, K. Hirose and N. Minematsu, "How to Improve TTS Systems for Emotional Expressivity", In Proc. of Interspeech, pp. 524–527, 2009.
9. J. R. W. Yi. "Natural-sounding speech synthesis using variable-length units". Master thesis, Massachusetts Institute of Technology, 1998.
10. A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In IEEE Tr. on Information Theory, 260–269, 1967.

11. A. Conkie, M.C. Beutnagel, A.K. Syrdal, and E. Philip. “Preselection of candidate units in a unit selection-based text-to-speech synthesis system”. In ICSLP, vol.3, 314–317, 2000.
12. R. Kumar. “A genetic algorithm for unit selection based speech synthesis”. In Proc. of INTERSPEECH, 2004.
13. P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. IEEE Transactions on System Science and Cybernetics, vol. 4, pp. 100–107, 1968.
14. N. J. Nilsson, “Principles of artificial intelligence”, Springer-Verlag editions, pp. 61-88, 1982.
15. Boeffard, O., Charonnat, L., Le Maguer, S., Lolive, D. and Vidal, G., “Towards fully automatic annotation of audiobooks for TTS”. In Proc. of LREC, 2012.
16. F. Alías, L. Formiga, and X. Llorá. “Efficient and reliable perceptual weight tuning for unit-selection text-to-speech synthesis based on active interactive genetic algorithms: A proof-of-concept”. Speech Communication, 53(5), 786–800, 2011.
17. R. E. Donovan. “A new distance measure for costing spectral discontinuities in concatenative speech synthesizers”. SSW4, 2001.
18. D. Cadic, C. Boidin, and C. D’Alessandro. “Vocalic sandwich, a unit designed for unit selection TTS”. In Tenth Annual Conference of the International Speech Communication Association (pp. 2079–2082), 2009.
19. D. Cadic, C. Boidin, and C. D’Alessandro. “Towards optimal TTS corpora”. In Proceedings of the Seventh International Conference on Language Resources and Evaluation, Valetta, Malta (pp. 99–104), 2010.