INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Recent Advances in Google Real-time HMM-driven Unit Selection Synthesizer

Xavi Gonzalvo, Siamak Tazari, Chun-an Chan,
Markus Becker, Alexander Gutkin, Hanna Silen

Google Inc., 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA
{xavigonzalvo,staz,cachan,mabecker,agutkin,silen}@google.com

Abstract

This paper presents advances in Google's hidden Markov model (HMM)-driven unit selection speech synthesis system. We describe several improvements to the run-time system; these include minimal latency, high-quality and fast refresh cycle for new voices. Traditionally unit selection synthesizers are limited in terms of the amount of data they can handle and the real applications they are built for. That is even more critical for real-life large-scale applications where high-quality is expected and low latency is required given the available computational resources. In this paper we present an optimized engine to handle a large database at runtime, a composite unit search approach for combining diphones and phrase-based units. In addition a new voice building strategy for handling big databases and keeping the building times low is presented.

Index Terms: speech synthesis, hybrid approaches, real-time, unit selection

1. Introduction

Recent advances in text-to-speech (TTS) synthesis research led to more intelligible and natural sounding synthetic speech than a decade ago. The quality of modern TTS systems mostly depends on the target domain (that can be limited or open) and particular deployment requirements, such as server-side applications or mobile devices.

The two dominant TTS design trends can be divided into concatenative and statistical approaches [1]. In this paper we present the salient features of Google's concatenative unit selection system used in many products and applications. These features were introduced in order to address the main challenges of unit selection synthesizers such as dealing with out-of-domain requests and handling large databases in real-time applications [2].

Our concatenation-based hybrid approach is based on the target prediction framework proposed in [3]. We are extending this framework with several improvements in different areas. First, we have optimized the voice building procedure to speed up the refresh cycle of new voices by an average of 65%. We also describe an optimized unit search algorithm using an extended set of features for better prosody modeling. Finally we introduce the improvements obtained by using longer units in order to deal with limited domain requests.

This paper is organized as follows. Section 2 describes the core of our unit selection system. Section 3 introduces the main features of our new real-time engine. Section 4 presents the improvements in the voice building system. Section 5 describes the approach for combining diphone and phrase-sized units. Section 6 describes our experiments and presents the results. Finally, Section 7 concludes the paper.

2. The core unit selection system overview

There are several types of hybrid systems that use HMM for unit selection. A full review of these can be found in [4]. Of particular interest to us are the two systems described in [3] and [5]. Similar to our approach, the system presented in [3] utilizes a composite target and join cost approach, defined using the HMM emission probabilities trained using the maximum-likelihood (ML) criterion. In [5] a unit selection system is presented which uses HMMs to constrain the prosodic parameters of target units.

Our unit selection system uses an HMM module to guide the selection of diphone-sized candidate units. It employs HMM emission probabilities trained using the ML criterion as target costs and the mel-frequency cepstral coefficient (MFCC) based ([6]) spectral differences for the join costs. The optimal preselected diphone sequence is expected to be chosen from the speech database to maximize the combined likelihood of the acoustic and diphone duration models.

For a target utterance $\mathbf{u}$ of length K, let $[u_1, u_2, \ldots, u_K]$ be a sequence of candidate units to realize it, and $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K]$ be the sequence of specification vectors, containing linguistic and prosodic contexts.

From a large set of possible units to concatenate, the best ones need to be chosen from the recorded database. The goal is to find a sequence $\mathbf{u}^*$ that minimizes the conventional unit selection cost [7] as described below.

Two cost functions are defined: the target cost $C_t(\mathbf{f}_k, u_k)$ is used to estimate the mismatch between the target specification vector $\mathbf{f}_k$ and the candidate unit u_k; the concatenation cost $C_c(u_k, u_{k+1})$ is used to estimate the smoothness of the acoustic signal when concatenating units u_k and u_{k+1}. Thus, the cost of a realization $\mathbf{u}$ is

$$C(\mathbf{u}, \mathbf{F}) = \sum_{k=1}^{K} C_t(\mathbf{f}_k, u_k) + \sum_{k=2}^{K} C_c(u_{k-1}, u_k)\,. \quad (1)$$

Unit selection can then be formulated as the problem of finding the optimal sequence of units $\mathbf{u}^*$ from candidate sequences $\mathbf{u}$ that minimizes the total cost,

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} C\left(\mathbf{u}, \mathbf{F}\right)\,. \quad (2)$$

Our system uses models which are derived from a set of units sharing the same linguistic features. For a target specification $\mathbf{f}_k$, the context-dependent acoustic model determined by clustered HMMs and decision trees is denoted by λ_k. The associated acoustic feature vector consists of a set of static and dynamic features. For a whole utterance, the corresponding target models are: $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_K]$.

Similarly, for each unit in the database, let $u_l = \{\lambda_i^{(b)}, \mathbf{o}_l\}$ refer to its observation vector $\mathbf{o}_l$ and to a contextual model from

http://dx.doi.org/10.21437/Interspeech.2016-264

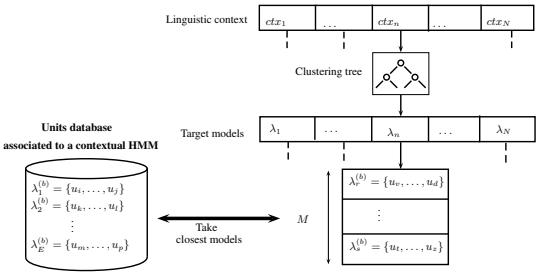

Figure 1: Diagram of the synthesis stage.

the database $\lambda_i^{(b)}$. Mapping between units and a corresponding HMM is given during alignment in the phoneme segmentation stage. Note that a single model will map to multiple units.

The process to find the optimal sequence of units is depicted in Figure (1) and works as follows.

Given an utterance, a target model is associated with its contextual diphone ($ctx_k \rightarrow \lambda_k$). This is obtained by clustering the linguistic contexts using a decision tree.

Using the target model λ_k for specification f_k, the M closest models are selected from the database $\{\lambda_{i_1}^{(b)}, \ldots, \lambda_{i_M}^{(b)}\}$. The purpose is to have several candidate models for the computation of target and join costs. We choose M significantly smaller than N, where N denotes the total number of models that exist in the database for the given diphone. This is because N can be large and thus storing the correspondences for all N models can be computationally prohibitive.

The closest M models are selected using a Kullback-Leibler (KL) divergence-based unit preselection algorithm proposed in [3]. The divergence $\mathrm{D}(\lambda_k, \lambda_i^{(b)})$ is computed between the HMM of the target model (λ_k) and the HMM of each candidate model $\{\lambda_{i_1}^{(b)}, \ldots, \lambda_{i_M}^{(b)}\}$. We then select the M-best models with minimum KL divergence before the final minimization process of the total cost. To minimize the run-time overhead the KL divergences are computed offline as a symmetric matrix for every two leaf nodes in the decision tree of each HMM state.

As the models under test share the transition matrix, the distance between HMMs can be simplified as

$$\mathrm{D}(\lambda_1, \lambda_2) \leq \sum_{i=1}^{S} \frac{1}{1 - a_{ii}} \left[\mathrm{d}\left(\mathcal{N}_{1_i}, \mathcal{N}_{2_i}\right) + \mathrm{d}\left(\mathcal{N}_{2_i}, \mathcal{N}_{1_i}\right) \right] , \tag{3}$$

where S is the number of states, a_{ii} is the self-transition probability for state i and d is the the KL divergence between two L-dimensional single mixture Gaussian distributions and can be calculated as in [3].

Using HMMs in a preselection algorithm offers two main advantages over traditional unit selection. First is the principled statistical modeling of the input linguistic feature space guided by the training data. As a consequence, this obviates the need for excessive manual hand-tuning of unit selection weights.

Good selection of linguistic features is critical for HMM-based unit selection algorithm. We use a large set of linguistic features. These include triphones as well as information at the syllable, word and phrase levels. These features distinguish phonological characteristics, prominence, phrasing, accent, syntax and intonation types. There are also positional features (e.g. number of syllables to previous stressed syllable in phrase) and syntactic dependencies (e.g. dependency depth or dependency size to right lower bound) [8].

3. Proposed engine optimizations

The main factors that impact quality of a unit selection system are the domain (which may be open or limited), database size and capacity of the engine to search the data during runtime. This section tackles the problem of how to preselect a large amount of units while maintaining a low latency. The advances that we are introducing in this paper are: 1. Using partial target cost computation within a heap to speed up unit preselection; 2. Significant speed ups in preselection by introducing the concepts of Gaussian Sequence Signatures (GSS) and computation order trees (COT); 3. Integer quantization of MFCC coordinates; and 4. Pruning certain paths during join cost calculations depending on the accumulated cost while searching.

3.1. Model selection

When a model is selected, units are taken in deterministic order and a linear search is conducted to compute the join cost. The classical alternative [3] is to do unit scoring (i.e. maximizing the likelihood of the units with respect to a target spectrum envelope) which completes the total target cost for each unit. While this solution may be adequate for finding the best units, it is not efficient performance-wise. We propose an admissible stopping criterion for the rapid selection of a subset of the units that have the highest probability of being adequate for the required context.

In our system all candidate models have to go through the full computation of equation (3), which contributes to $N \cdot S \cdot R$ calls (N being the number of models in this diphone, S the number of states and R the number of streams). However D can be decomposed into multiple steps, say, $\mathrm{D}_1 \rightarrow \mathrm{D}_2 \rightarrow \ldots \mathrm{D}_S = \mathrm{D}$, such that they contribute to the same amount of computations of D and their values are non-decreasing, so $\mathrm{D}_1 \leq \mathrm{D}_2 \leq \cdots \leq \mathrm{D}_S = \mathrm{D}$.

As D is a summation of d's and the latter is nonnegative, we can define the recursion as

$$\begin{aligned} \mathrm{D}_s(\lambda_1, \lambda_2) &= \sum_{i=1}^{s} \mathrm{d}_{state}(\lambda_1, \lambda_2, i) \\ &= \mathrm{D}_{s-1}(\lambda_1, \lambda_2) + \mathrm{d}_{state}(\lambda_1, \lambda_2, s) \,. \end{aligned}$$

Since D_s is non-decreasing, it can be treated as a lower bound of the final distance D. We maintain a min-heap of partial distances, iteratively update (from D_s to D_{s+1}) the top item (whose partial distance is minimum at the moment) and adjust its position in the heap. Whenever a top item is completed, we pop it from the heap and check if enough units have been gathered. If this condition is met, we stop the iteration. This algorithm is correct because when an item is popped, its final distance is already smallest compared to the partial distance of all other items still in the tree.

Each update or popping from the root of the heap requires $O(\log N)$ time so we are in the worst case using $O(S \cdot N \log N)$ time. In practice, the number of calls to d is much smaller than $N \cdot S \cdot R$.

3.2. Gaussian sequence signatures

One potential speedup of the above algorithm is to aggregate the identical computation procedures for different models. We define the GSS of a model to be the sequence of Gaussian integer identifiers (GIDs) in all the states and streams of that model. Some GIDs are widely shared among models in the same diphone; this provides us the opportunity to reduce the unnecessary computations. For example, the number of identical GSS

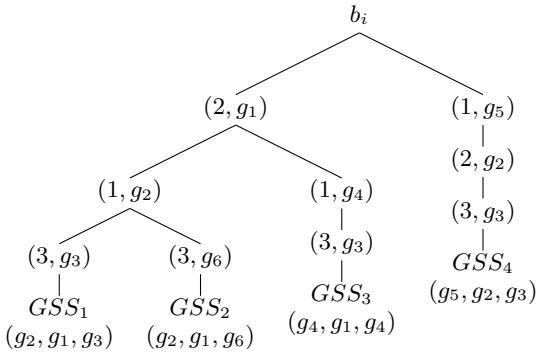

Figure 2: *An example COT for three state HMMs to compute* D *for four GSS' in diphone* b_i*. The tuple in each node represent (state index, Gaussian index) where stream ID is omitted. The actual GID is listed at the bottom of the leaf nodes.*

is only 35.07% of the number of models for diphone "ax-n" in our database. The simplest idea is to group all models with the identical GSS together, and copy the score from the GSS to the model after the computation is finished. We make a map from GSS to model set, and replace GSS during the computation of the KL divergence and expand the set of the models after the GSS is selected. This way we can substantially reduce both the number of calls to d and the heap size.

To take this idea further, we can share partial distance calculation for all models having the same partial GIDs, which is directed by a COT. Each node in a COT defines a summand in equation (3) to be added to the partial distance. Traversing a path from root to the leaf node will give us the order of computation for a GSS. For example, assuming we have only one stream and three states, diphone b_i could contain all the Gaussian sequences depicted in Figure 2. Three GSS' in b_i have their GID in state 2 to be g_1. They are aggregated for only one computation. We can regard the node $(2, g_1)$ as a collection of GSS' which match $(*, g_1, *)$. Similarly $(1, g_2)$ represents GSS' matching $(g_2, g_1, *)$. Instead of GSS, we put nodes of the COT into the heap (initially the root node). Whenever we pop the node from the heap, we advance the computation suggested by it's children, and push them into the heap.

It is worth noting that when using the regular total cost defined in equation (1), we can skip KL computation for d if all the GSS in the diphone share the same GID because that distance component is constant for all candidates.

3.3. Quantization and Pruning

To further optimize the computation of join costs we use integer quantization, taking advantage of the dynamic range of MFCC values. We established in listening tests that integer quantization did not affect quality.

In addition, we apply an admissible stopping criterion in the local minimization of the unit selection Viterbi computation [9]. By sorting the results of the previous time step in ascending order of path cost, we can abandon the minimization of the total path cost for the current time step with no approximation error whenever the path cost of the previous column is higher than the current best result.

4. Voice building system

During voice building the following steps take place. First, phone-level alignment is performed to obtain the phoneme boundaries. This step uses phone HMMs trained over all utterances and is relatively fast compared to the rest of the process. We then perform diphone alignment that merges the half-phone time boundaries from the previous step. Next, given the diphone boundaries, the full-context HMM training involving expectation-maximization (EM) and decision tree clustering takes place [10]. In our left-to-right five states system each HMM has different types of information in each model (spectrum, fundamental frequency and aperiodicity), so in total there are fifteen trees plus a duration model tree. Finally, the last step pre-computes the score matrices containing KL divergences among all leaf nodes of all decision trees.

The critical voice building part is the HMM training and here we propose an alternative approach that preserves the quality of the system while dramatically reducing the execution time. As described in section 2, the HMMs are used in the voice for scoring purposes and not for parameter generation. Hence we hypothesize that target costs are less dependent on the state boundaries than speech reconstruction is. Therefore training can be simplified in order to keep the Gaussian statistics intact assuming that state boundaries are in fact not that critical. Regular HMM training has two iterations of the following steps: (i) Viterbi initialization and re-estimation where the state alignment is fixed. (ii) Embedded re-estimation where state alignment is modified. (iii) Duration model re-estimation is performed jointly with the rest of the models in the case of Hidden Semi-Markov Models (HSMM) [11]. (iv) Decision tree building using first and second order statistics.

Maximum likelihood estimates of the HMM state parameters are given using the probability that an observation (i.e. input frame of acoustic data) was produced by mixture m of state j. This would normally be calculated by the forward-backward algorithm during the EM re-estimation. What we propose is to avoid the state re-estimation and simply compute the statistics of those models assuming that the state boundaries are fixed. This assumption dramatically simplifies the computation.

Since decision tree building uses state occupancy counts, first and second-order statistics, we need to generate those based solely on the number of examples. In addition, a large number of models means that fewer samples will be available for estimating each model, negatively impacting variance estimation. The quality of variances has a direct impact on the size of decision trees and hence on the overall quality of the synthesizer. For this reason the use of variance flooring was introduced into the system along with unsmoothed F_0 contours.

5. Long units

There are basically two ways of achieving the ultimate unit selection synthesizer with respect to the balance between naturalness and flexibility. On the one hand, the flexible approach aims at producing natural speech synthesis by selecting small units (e.g. diphones). On the other hand, the rigid approach seeks to maintain naturalness by working in a constrained domain. The main advantage of the rigid approach is the possibility of using longer units (e.g. phrases) as the main concatenative unit.

An approach for long units was proposed in [12] where phrase-sized units were taken intact from a large corpus. This is shown to be adequate to express paralinguistic information which otherwise would be very difficult to synthesize since it would imply modifications of prosody and voice quality. Another alternative is to use concept to speech (CTS) synthesis integrating the natural language generation (NLG) and the synthesizer so that they share the same features [13].

Similarly, primitive synthesizers such as [14, 15, 16] prior-

itized pre-recorded prompts in a template-based synthesis system in order to minimize joins and maximize expressiveness.

We are revisiting those approaches to create a synthesizer with non-uniform units favoring the longer ones and backing-off to shorter ones when needed. To identify longer units we propose to use a set of labels to retrieve the metadata of the message. For this we are labeling the data with hints about the content of the sentence [17] and its prosody. We also integrate phrase-sized units into the lattice by using a trie data structure. Phrase-sized chunks are treated as slightly favored candidate units in addition to the normal units chosen by preselection; this way, the Viterbi search will determine the best points to transition into and out of the template using generic units.

6. Experiments

We present mean opinion score (MOS) and AB tests. A total of 173 test utterances were used. One subject could evaluate a maximum of 30 stimuli. Each pair was evaluated by five subjects in the AB test and by three subjects in the MOS tests. Subjects used headphones.

6.1. Optimized engine

The new engine proposed in the previous sections reduced the latency and gave the system the possibility of handling a larger database during runtime. Ten-fold analysis of the required processing time as a function of search unit count for US-English is presented in Figure 3. Results for the previous baseline system (blue) and the optimized system (green) indicated the possibility of using three times more search units with the optimizations.

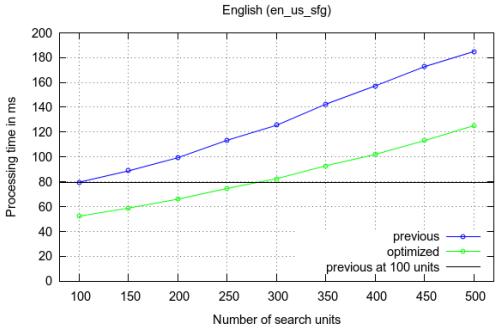

Figure 3: *Processing time in ms (average of ten-fold runs).*

Table 1 shows the MOS for US-English and French synthesis reached by all the optimizations to the engine while maintaining a low latency. For both languages, the new system involved 20% more linguistic features and three times more search units compared to the baseline.

Table 1: *MOS for new optimized engine.*

Language	Baseline	Proposed
US-English	3.61 ± 0.1	$\mathbf{3.82 \pm 0.08}$
French	3.68 ± 0.11	$\mathbf{3.85 \pm 0.106}$

6.2. Long units

The impact of merging diphone and phrase-sized units was noticeable for specific TTS applications. For US-English, the overall effect of using longer units, as evaluated in an AB comparison test on in-domain lines, resulted in 0.331 ± 0.083 preference for the long units approach ($p < 10^{-6}$); on a mixed test set, the preference was 0.095 ± 0.042.

Furthermore, Table 2 showed the degradation MOS (DMOS) quality evaluation results for US-English analyzing a total of four domains (Domain I-IV), using both the long-units technique and a revised database that is catered to this approach, but is not larger in total size. Domain I and III contained a high amount of labeled information and as we can see, using phrase-sized units had a significant positive impact. It is worth noting that Domain IV consisted mainly of out-of-domain lines but still enjoyed an improvement.

Table 2: *DMOS test results for US-English with long units.*

Domain I	Domain II	Domain III	Domain IV
+0.65	+0.47	**+0.52**	+0.25

6.3. Voice building

The objective of the voice building experiments was to validate the new proposed strategy in terms of speed and quality degradation. Table 3 contains the comparison of the time required to build a voice for US-English, German, and Spanish.

Table 3: *Building time in minutes.*

Language	Baseline	Fast	Speed-up (Percentage)
US-English	1140	**126**	80.1%
German	849	**238**	56.2%
Spanish	453	**95**	65.3%

In order to validate that the synthesis quality was preserved regardless of the optimization in the voice building process, results from two AB comparison tests are presented. Table 4 shows the result for the AB test comparing fast voice building with 3-5 states against the regular voice building for US-English. Four-state fast voice building and regular voice building showed no significant preference. Language-wise AB comparison for US-English, German, and Spanish further validated the new proposed approach. The results presented in Table 5 showed no significant preference towards any of the two systems.

Table 4: *AB test results for fast US-English voice building.*

States	Score	p-value	Preference
5	-0.169 ± 0.151	0.00301	Regular
4	-0.075 ± 0.148	0.350	**No preference**
3	-0.160 ± 0.148	0.00340	Regular

Table 5: *AB test results for US-English, German, and Spanish.*

Language	Score	p-value	Preference
US-English	0.020 ± 0.155	0.823	No preference
German	0.043 ± 0.184	0.777	No preference
Spanish	0.084 ± 0.110	0.0759	No preference

7. Conclusions

In this paper we have presented new improvements to Google's unit selection synthesizer aimed at handling large amounts of data during both runtime and voice building time. We have presented a new voice building strategy that reduced the total building time dramatically and we have shown to maintain quality. We have also presented a novel approach to cope with a large database during runtime reducing latency and increasing the quality. Finally we have shown an algorithm to merge diphone and phrase-sized units in a single framework to increase the quality of the limited domain applications.

8. References

[1] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[2] J. Schroeter, A. Conkie, A. Syrdal, M. Beutnagel, M. Jilka, V. Strom, Y. J. Kim, H.-G. Kang, and D. Kapilow, "A perspective on the next challenges for TTS research," in *Speech Synthesis, 2002. Proceedings of 2002 IEEE Workshop on*. IEEE, 2002, pp. 211–214.

[3] Z. Ling, L. Qin, H. Lu, Y. Gao, L. Dai, R. Wang, Y. Jiang, Z. Zhao, J. Yang, J. Chen, and G. Hu, "The USTC and iFlytek speech synthesis systems for Blizzard Challenge 2007," in *Proc. of Blizzard Challenge Workshop*, 2007.

[4] X. Gonzalvo, A. Gutkin, J. C. Socoró, I. Iriondo, and P. Taylor, "Local minimum generation error criterion for hybrid HMM speech synthesis," in *Proc. of ICSLP*, Brighton, UK, 2009, pp. 416–419.

[5] H. Kawai, T. Toda, J. Ni, M. Tsuzaki, and K. Tokuda, "XIMERA: A new TTS from ATR based on corpus-based technologies," in *Proc. of the IEEE Speech Synthesis Workshop*, 2004, pp. 179–184.

[6] F. Zheng, G. Zhang, and Z. Song, "Comparison of different implementations of MFCC," *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 582–589, 2001.

[7] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 373–376.

[8] R. T. McDonald, J. Nivre, Y. Quirmbach-Brundage, Y. Goldberg, D. Das, K. Ganchev, K. B. Hall, S. Petrov, H. Zhang, O. Täckström *et al.*, "Universal Dependency Annotation for Multilingual Parsing." in *ACL (2)*, 2013, pp. 92–97.

[9] S. Sakai, T. Kawahara, and S. Nakamura, "Admissible stopping in Viterbi beam search for unit selection in concatenative speech synthesis," in *Proc. of ICASSP*, 2008, pp. 4613–4616.

[10] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Proc. of Eurospeech*, 1999, pp. 2347–2350.

[11] H. Zen, K. Tokuda, T. Masuko, T. Kobayasih, and T. Kitamura, "A Hidden Semi-Markov Model-Based Speech Synthesis System," *IEICE Trans. Inf. Syst.*, vol. E90-D, no. 5, pp. 825–834, 2007.

[12] N. Campbell, "Conversational speech synthesis and the need for some laughter." vol. 14, no. 4, 2006, pp. 1171–1178.

[13] P. A. Taylor, "Concept-to-speech synthesis by phonological structure matching," in *Philosophical Transactions of the Royal Society, Series A*, 2000, pp. 623–626.

[14] E. Klabbers, "High-quality speech output generation through advanced phrase concatenation," in *In Proceedings of the COST Workshop on Speech Technology in the Public Telephone Network: Where are We Today*, 1997, pp. 85–88.

[15] L. Gauvain, L. F. Lamel, J. L. Gauvain, B. Prouts, C. Bouhier, and R. Boesch, "Generation and synthesis of broadcast messages," in *Proc. ESCA-NATO Workshop on Applications of Speech Technology, Lautrach*, 1993, pp. 207–210.

[16] R. E. Donovan, M. Franz, J. S. Sorensen, and S. Roukos, "Phrase splicing and variable substitution using the IBM trainable speech synthesis system," in *Proceedings of the 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '99, Phoenix, Arizona, USA, March 15-19, 1999*, 1999, pp. 373–376.

[17] F. Alías, X. Sevillano, J. C. Socoró, and X. Gonzalvo, "Towards high quality next-generation Text-to-Speech synthesis: a Multidomain approach by automatic domain classification," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 7, pp. 1340–1354, 2008.