

# DATA-DRIVEN SEGMENT PRESELECTION IN THE IBM TRAINABLE SPEECH SYNTHESIS SYSTEM

*Wael Hamza<sup>1</sup> and Robert Donovan<sup>2</sup>*

IBM T.J. Watson Research Center,  
Yorktown Heights, NY, 10598, USA

<sup>1</sup> hamzaw@us.ibm.com

<sup>2</sup> red@watson.ibm.com

## ABSTRACT

Unit selection based concatenative speech synthesis has proven to be a successful method of producing high quality speech output. However, in order to produce high quality speech, large speech databases are required. For some applications, this is not practical due to the complexity of the database search process and the storage requirements of such databases. In this paper, we propose a data-driven algorithm to reduce the database size used in concatenative synthesis. The algorithm preselects database speech segments based on statistics collected by synthesizing a large number of sentences using the full speech database. The algorithm is applied to the IBM trainable speech synthesis system and the results show that database size can be reduced substantially while maintaining the output speech quality.

## 1. INTRODUCTION

Concatenative speech synthesis using large speech databases has become popular due to its ability to produce high quality speech output [1,2,3,4,5,6]. Typically, a large amount of speech data is recorded and processed to produce the synthesis unit inventory of speech segment waveforms and associated information. During synthesis, the input text is converted to a string of phonetic and prosodic targets. The synthesis system then searches for speech segments that are close to the required target string while trying to maintain spectral continuity between adjacent segments. This is achieved using a dynamic programming algorithm in conjunction with an appropriate cost function. The selected segments are then concatenated to produce the synthetic speech.

In order to produce high quality speech, a large amount of speech is used to construct the synthesis database. This makes it more likely that the synthesis system will find segments that are very close to the targets, and so reduces or eliminates the need for any signal modification that would otherwise reduce the speech quality. However, using a large speech database is not practical for many applications due to the storage requirement and the computational complexity involved in searching it.

In the IBM trainable speech synthesis system, [1,2], many algorithms have been used to reduce the size of the speech database including simple algorithms [1] and complex ones [7]. Although the size of the database is reduced using these algorithms, a noticeable degradation in synthesis quality has been observed. All algorithms used to date relied only on

information extracted from the database, and not on the behavior of the database during synthesis. For example, the simple algorithm mentioned in [1] selects a large number of contiguous segments by retaining the first N segments in each decision tree leaf.

In this paper, we present a data-driven algorithm to reduce the database size. The algorithm uses statistics collected by synthesizing a large number of sentences to preselect speech segments. Unlike previous algorithms, the proposed algorithm relies on the behavior of the speech database during synthesis. The approach is similar to that followed in [8].

This paper is organized as follows. Section 2 briefly describes the IBM trainable speech synthesis system in which the proposed algorithm is used. Section 3 describes the proposed algorithm in detail. Listening test results and real time performance results are presented in section 4.

## 2. IBM TRAINABLE SPEECH SYNTHESIS SYSTEM OVERVIEW

During the speech database building process, 1400 – 2000 sentences of training script are recorded providing 2.5-3.5 hours of speech data. The recordings are made at a 22kHz or 44kHz sampling rate. The training script is generated from a large number of sentences (approximately 100,000) using a greedy algorithm to select sentences containing as much variety in phonetic context as possible. A laryngograph signal is recorded with the speech signal and is processed to determine the instants of glottal closure in regions of voiced speech. This information is used by the segment concatenation/modification module in synthesis.

The speech data is coded into 12 dimensional mel-frequency cepstral coefficients plus log energy and the first and second derivatives of these parameters. The speech data is then aligned using a set of speaker independent hidden Markov models (HMMs). The resulting alignment is used to train speaker-dependent decision-tree state-clustered HMMs. These HMMs are used to obtain the final alignment of the speech data. The final alignment is used to build acoustic decision trees for each unclustered HMM state using the standard maximum likelihood tree building procedure from speech recognition [9]. The speech segments belonging to each leaf in the resulting trees are kept together with information about their energy, pitch, duration and spectral endpoints. Energy prediction decision trees are also built for each unclustered HMM state using the same tree building algorithm. Spectral continuity cost decision trees are

built using the final alignment as described in [10]. These trees are used to determine the spectral continuity cost between speech segments during the runtime segment search in synthesis.

During synthesis, text processing, text to phone conversion, and prosody prediction are performed using an independent rule-based front-end. The resulting phonetic sequence is used to traverse the acoustic decision trees to give the corresponding synthesis leaf sequence. Target energies are determined for each leaf using the energy decision trees. A target duration is determined for each leaf as the median duration of the segments in the acoustic leaf scaled such that the sum of the leaf durations in each phone is equal to the target phone duration predicted by the front-end. A search is performed over the speech segments in each leaf of the synthesis leaf sequence to obtain the segment sequence that is closest to the prosodic targets while minimizing the spectral continuity cost between adjacent segments<sup>1</sup>. This is done using a cost function and dynamic programming strategy similar to that in [11]. The continuity cost is computed from the perceptually modified cepstral vectors and continuity cost trees described in [10]. The resulting segment sequence is passed to the segment modification/concatenation module, which concatenates the segments after modifying their prosody to match the target prosody. More details about the current status of the synthesis system can be found in [1].

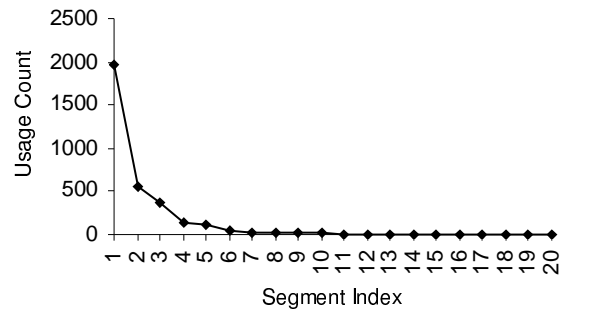
### 3. THE PROPOSED ALGORITHM

As mentioned in the previous section, a dynamic programming search is performed to obtain the speech segments closest to the required targets while minimizing the spectral discontinuity between adjacent segments. With very large databases, many leaves, especially those corresponding to commonly used contexts, may contain large number of segments. This slows down the search substantially due to the increased number of cost function evaluations which must be performed, especially when two consecutive leaves both contain large numbers of segments. Although the training script is generated using an automatic algorithm, the resulting script contains natural sentences that have a lot of redundancy in terms of phonetic and prosodic features. One possible way of reducing the database size is to try to get rid of this redundancy by reducing the number of similar segments in a particular acoustic leaf. In addition, some segments may not be useful due to their outlying prosodic features. For example, some segments could be labeled with incorrect duration and/or pitch values due to misalignment and/or pitch determination problems respectively.

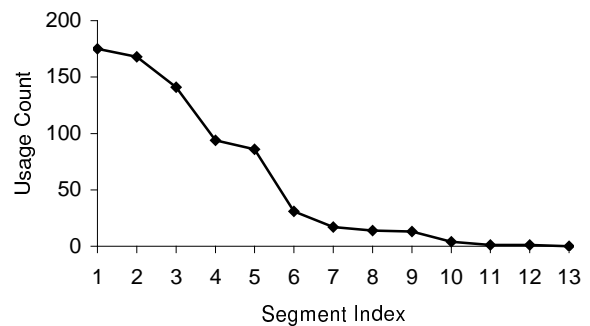
Database size could be reduced by detecting and eliminating such redundancy and discarding outlying segments. The proposed algorithm relies on the synthesizer behavior at runtime to identify such segments. A large number of sentences (approximately 100,000) is synthesized using the full speech database and segment usage statistics are collected. The usage statistics are a good measure of how useful each segment is to the synthesizer. Discarding the least useful segments from the database reduces the database size and speeds up the search

<sup>1</sup> In [1], segments belonging to other acoustic leaves in the same unclustered-state tree as each target leaf are also considered during the search, for a cost penalty. However, in the current work, this was not allowed.

substantially. The discarded segments are likely to be either redundant, outlying, or segments belonging to rare contexts. While discarding redundant and outlying segments is desirable, discarding those in rare contexts may not be. Although individual contexts may be extremely rare, the combined probability mass of all rare contexts may be sufficiently large that the omission of such segments may cause problems during synthesis [12]. In this work, no distinction was made between these different types of rarely used segments. Making this distinction is a subject for possible future research. Fig. 1 shows segment usage statistics for two different acoustic leaves. The segments are sorted by usage count to create a meaningful display.



(a)



(b)

Figure 1. Usage statistics for segments in two leaves. The horizontal axis represents segment index in the leaf under consideration while the vertical axis represents the number of times the segment was used during statistics collection. Segments on the horizontal axis are sorted by usage count to create a meaningful display.

It is clear from Fig. 1-a that some of the segments in the displayed acoustic leaf are used frequently while other segments are rarely or never used. In the unselected system, all segments are included in the search each time this acoustic leaf is used. This could be considered a waste of time and storage. In Fig. 1-b, usage is more evenly distributed over the segments. It is clear that the leaf displayed in Fig. 1-a was used more frequently than the one displayed in Fig. 1-b when

collecting statistics. It was found that this more-even usage distribution occurred mostly in the least used leaves.

The proposed algorithm preselects segments as follows. Each acoustic leaf is considered in turn. All segments that belong to the leaf under consideration are sorted based on usage statistics as shown in Fig. 1. Then, starting from the most used segment, segments are labeled preselected and their usage statistics are accumulated. The algorithm stops when the accumulator reaches a specified fraction of the total usage of that leaf. A minimum number of segments per leaf is enforced in order to make the algorithm robust for targets not seen in the statistics collection phase. A maximum number of segments per leaf is optionally enforced.

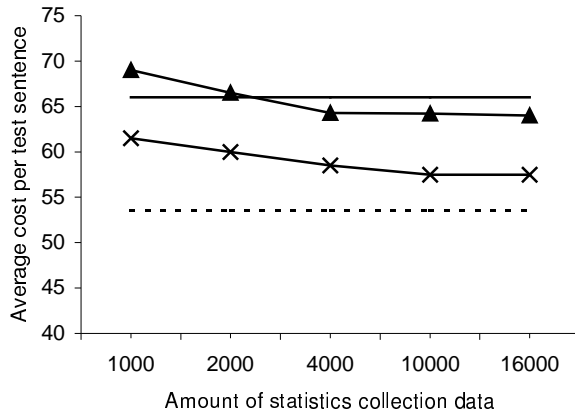


Figure 2. Average cost vs. amount of data used to collect statistics. The horizontal axis represents the number of sentences used to collect usage statistics. The vertical axis represents the average cost of synthesizing 2000 test sentences. The horizontal dotted line represents the unpreselected system. The horizontal solid line represents the simple preselection algorithm described in section 1 when used to preselect 50% of the unpreselected database. The solid line with x marks represents the proposed algorithm used to preselect 50% of the unpreselected database. The solid line with  $\Delta$  marks represents the proposed algorithm used to preselect 30% of the unpreselected database.

An experiment was conducted to determine the amount of data required for usage statistics collection. Datasets were preselected based on increasing amounts of data, and each used to synthesize a set of 2000 test sentences. The test sentences were neither in the training script nor in the data used for statistics collection. The average cost per sentence was calculated by averaging the total dynamic programming cost for synthesizing all the test sentences. The results of the experiment are shown in Fig 2. Although the graph suggests that the algorithm could stop collecting statistics after 10,000 sentences, we decided to collect statistics from as much data as is practical because the cost function may not be a good stopping criterion [7]. It is also clear from the figure that the proposed algorithm outperforms the simple preselection algorithm described in section 1 at least in terms of average synthesis cost.

## 4. RESULTS

In order to assess the proposed algorithm, a listening test was performed. A system was built using a female voice and the procedure described in section 2. The system had 10312 leaves and 159830 non-silence segments<sup>2</sup>. The trees were built to have a minimum of 10 segments per leaf. The listening test was performed on the following three systems.

- **Unpreselected:** This system used the full speech segment database, containing 159830 non-silence segments.
- **Simple:** This system used the simple preselection algorithm described in section 1. It retained the first 10 segments in each of the leaves. This system contained 93307 non-silence segments.
- **Proposed:** This system used the new preselection algorithm proposed in section 3. It contained 93294 non-silence segments in order to be comparable with the simple preselected system. This number was achieved by using a minimum of 1 segment per leaf, a maximum of 100, and a usage fraction of 99.803%.

A set of news sentences was used as the test material. These sentences were neither in the training script nor in the data used to collect statistics when using the proposed algorithm. Ten different random sets were prepared each containing 48 test sentences, 16 of which were synthesized by each system being tested. The ten sets were played to ten listeners who were asked to rate each sentence on a scale of 1 (bad) to 5 (good). The listeners were asked to give the rating based on how good they thought the sentence was without any further definition of the word “good”. Before starting the test, the listeners were played the first ten sentences to give them an idea of the speech quality they were about to listen to. They were encouraged to use the full 5-point scale. The score for each system was therefore evaluated as an average of 160 data points. The scores are displayed in Fig. 3.

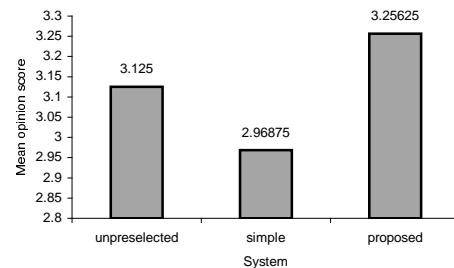


Figure 3. Mean opinion score for the unpreselected system, the simple preselected system, and the proposed preselected system.

The unpreselected system was not significantly better or worse than either of the preselected systems using one-tailed t-tests at the 5% level. However, we found that the proposed

<sup>2</sup> Silence segments are not used in synthesis.

algorithm was significantly better than the simple preselection algorithm, using a one-tailed t-test at the 5% level.

Fig. 4 shows synthesis time for the two preselected systems relative to the unpreselected one. Relative synthesis times were obtained by synthesizing 100 long sentences using the three systems. It is clear from the figure that the proposed preselected system runs 2.2 times faster than the unpreselected system and nearly as fast as the simple preselected system. Similar speed improvement results were obtained in [8] also with no significant difference in mean opinion score between the preselected and unpreselected systems.

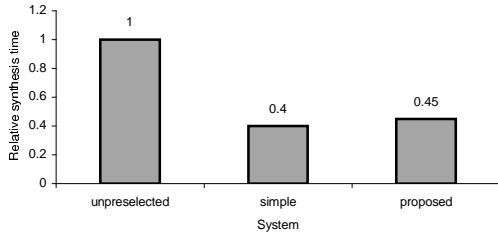


Figure 4. Synthesis time relative to the unpreselected system for the unpreselected system, the simple preselected system, and the proposed preselected system.

## 5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a data-driven algorithm to reduce the speech database size in decision tree based concatenative speech synthesis systems. The algorithm was applied to the IBM trainable speech synthesis system. A substantial reduction in the database size was achieved while maintaining the speech quality. This results in an improvement in the system's synthesis speed. Listening tests have shown that the proposed algorithm performs better than the simple preselection algorithm used previously. The proposed algorithm may prove to be a very useful tool in future experiments involving larger amounts of training data aimed at improving synthesis quality. Possible areas of future research include the investigation of methods to distinguish between potentially useful segments with rare contexts and those which are redundant or outlying due to mislabeling.

## 6. REFERENCES

[1] Donovan, R.E., Ittycheriah, A., Franz, M., Ramabhadran, B., Eide, E., Viswanathan, M., Bakis, R., Hamza, W., Picheny, M., Gleason, P., Rutherford, T., Cox, P., Green, D., Janke, E., Revelin, S., Waast, C., Zeller, B. Guenther, C. & Kunzmann, J. (2001) "Current Status of the IBM Trainable Speech Synthesis System," Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Atholl Palace Hotel, Scotland<sup>3</sup>.

[2] Donovan, R.E., and Eide, E.M. (1998) "The IBM Trainable Speech Synthesis System," Proc. ICSLP'98, Sydney.

[3] Syrdal, A. K., Wightman, C. W., Conkie, A., Stylianou, Y., Beutnagel, M., Schroeter, J., Strom, V., Lee, K., Makashay, M. J. (2000) "Corpus-Based Techniques in the AT&T NextGen Synthesis System," Proc. ICSLP'00 Beijing, China.

[4] Coorman, G., Fackrell, J., Rutten, P., Van Coile, B. (2000) "Segment Selection in the L&H RealSpeak Laboratory TTS System," Proc. ICSLP'00, Beijing, China.

[5] Hamza, W., Rashwan, M. (2000) "Arabic Concatenative Speech Synthesis Using Large Speech Database," Proc. ICSLP'00, Beijing, China.

[6] Quazza, S., Donetti, L., Moisa, L., Salza, P.L. (2001) "Actor: A Multilingual Unit-Selection Speech Synthesis System," Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Atholl Palace Hotel, Scotland<sup>3</sup>.

[7] Donovan, R.E. (2000) "Segment Pre-selection in Decision-Tree Based Speech Synthesis Systems," Proc. ICASSP'00, Istanbul.

[8] Conkie, A., Beutnagel, M. C., Syrdal, A. K., Brown, P. E. (2000) "Preselection of Candidate Units in a Unit Selection-Based Text-to-Speech Synthesis System," Proc. ICSLP'00, Beijing, China.

[9] Bahl, L.R., deSouza, P.V., Gopalakrishnan, P.S., and Picheny, M.A. (1993) "Context Dependent Vector Quantization for Continuous Speech Recognition," Proc. ICASSP'93, Minneapolis.

[10] Donovan, R.E. (2001) "A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesizers," Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Atholl Palace Hotel, Scotland<sup>3</sup>.

[11] Hunt, A., Black, A. (1996) "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database," Proc. ICASSP'96, Atlanta.

[12] Mobius, B. (2001) "Rare Events and Closed Domains: Two Delicate Concepts in Speech Synthesis," Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Atholl Palace Hotel, Scotland<sup>3</sup>.

<sup>3</sup> Available online from <http://www.ssw4.org>