INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Siri On-Device Deep Learning-Guided Unit Selection Text-to-Speech System

Tim Capes, Paul Coles, Alistair Conkie, Ladan Golipour, Abie Hadjitarkhani, Qiong Hu, Nancy Huddleston, Melvyn Hunt, Jiangchuan Li, Matthias Neeracher, Kishore Prahallad, Tuomo Raitio, Ramya Rasipuram, Greg Townsend, Becci Williamson, David Winarsky, Zhizheng Wu, Hepeng Zhang*

Apple Inc., USA

Abstract

This paper describes Apple's hybrid unit selection speech synthesis system, which provides the voices for Siri with the requirement of naturalness, personality and expressivity. It has been deployed into hundreds of millions of desktop and mobile devices (e.g. iPhone, iPad, Mac, etc.) via iOS and macOS in multiple languages. The system is following the classical unit selection framework with the advantage of using deep learning techniques to boost the performance. In particular, deep and recurrent mixture density networks are used to predict the target and concatenation reference distributions for respective costs during unit selection. In this paper, we present an overview of the run-time TTS engine and the voice building process. We also describe various techniques that enable on-device capability such as preselection optimization, caching for low latency, and unit pruning for low footprint, as well as techniques that improve the naturalness and expressivity of the voice such as the use of long units.

Index Terms: Speech synthesis, unit selection, hybrid, recurrent mixture density network, on-device

1. Introduction

Text-to-speech (TTS) synthesis is an essential component of a voice-based intelligent personal assistant (e.g. Siri) and other voice interactive applications (e.g. navigation). The goal of a TTS system is to produce highly intelligible, expressive and natural-sounding synthetic speech that is indistinguishable from human speech.

There are two mainstream techniques for industry production development, namely waveform concatenation (i.e. unit selection) and statistical parametric speech synthesis (SPSS) [1]. Given a sequence of text input, unit selection directly assembles waveform segments to produce synthetic speech, while SPSS predicts synthetic speech from trained acoustic models. Unit selection typically produces more natural-sounding speech than SPSS, provided the database used has sufficient high quality audio material.

Unit selection synthesis in its current form has its origin with [2]. Unit sizes are most often chosen to be half phones or diphones or sometimes demi-syllables [3]. Normally, minimal signal processing is done in such a system. The system performs best when the database is large, and when the audio quality is good. For commercial systems, professional voice actors are often selected as the voice talent(s).

Most recently, much work has centered on using a statistical model to predict acoustic and prosodic parameters for synthesis and then using these predictions to set the costs in a unit selection system – this is known as hybrid unit selection [4]. A variety of techniques have been studied for hybrid unit selection including recent works that use deep learning techniques [5, 6, 7]. Our system is following this direction and employing deep learning techniques to implement both concatenation and target costs to improve unit selection.

In this paper, we present an overview and implementation details of the Apple Siri unit selection speech synthesis system, which has been deployed into hundreds of millions of desktop and mobile devices (e.g. iPhone, iPad, Mac, etc.) through iOS and macOS. In particular, we have the following contributions: first, we use deep and recurrent mixture density networks (MDNs) to predict target and concatenation distributions and jointly implement the target and concatenation costs in a probabilistic way; second, we introduce multiple optimizations (e.g., long units, preselection, and unit pruning) for naturalness, low latency, and low footprint; and third, we also describe our language-neutral voice building process.

2. Hybrid Unit Selection System Overview

Our on-device TTS system follows the typical unit selection framework, which uses a front-end to produce linguistic features, pre-selection for low latency, statistical model to implement concatenation and target costs for Viterbi search that finds the optimum unit sequence, and waveform concatenation to generate the final synthesized waveform. As a hybrid system, it benefits from a unified deep learning-based acoustic model to predict acoustic and prosodic feature distributions and to implement concatenation and target costs. The reason we choose this framework is that it can produce higher quality than statistical parametric speech synthesis with acceptable footprint and latency. Additionally as an on-device system, it allows synthesis without an internet connection. In this section, we will briefly introduce each module and some additional optimizations for low-latency and quality.

2.1. Front-end

The first step in synthesis is to process input text and generate phonetic transcription. The main goal of the front-end (also known as text processing), is to generate a phonetic transcription of the raw input text alongside several linguistic features (punctuation, syllabification, accentuation) in order to guide the prosody prediction and unit selection steps to produce intelligible and natural speech. To supplement the natural features of the input text, the front-end can also incorporate explicit annotations placed in the text stream to provide hints about pacing, prosody, and discourse domain, many of which are known for specific types of Siri responses. We are currently using a tradi-

*Authors listed in alphabetical order by the last name.

http://dx.doi.org/10.21437/Interspeech.2017-1798

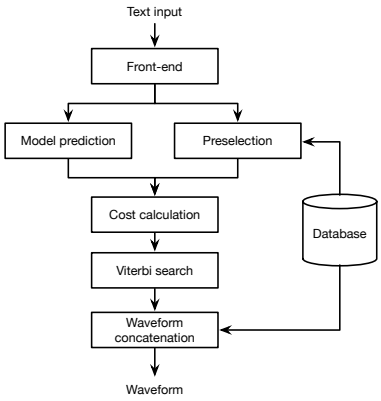

Figure 1: *An overview of the Siri on-device deep learning-guided unit selection speech synthesis engine. In our system, there is no dependency between preselection and model prediction which use deep and recurrent neural nets to predict target and concatenation distributions for cost calculation, and hence the tasks can be swapped or performed in parallel in the practical implementation.*

tional and largely rule-based system to generate the necessary linguistic features.

2.2. Preselection

The devices on which our synthesis system runs are fairly memory constrained and consequently the operating system enforces much stricter memory usage policies than one would encounter in a typical contemporary desktop or server environment. Consequently, our system relies extensively on read-only memory mapped data (which can be managed efficiently by the OS), and tries to minimize the size of resident data needed to perform unit selection. This is accomplished by constraining the set of candidate units for unit selection through a two-step preselection.

Phoneme context matching: As a first pass, we identify a set of unit candidates for each half phone that matches the target phonetic context as accurately as possible. We first consider the subset of candidates that are a quinphone match for the desired context, and then augment with subsets of triphone, diphone, and finally monophone matches until we have met or exceeded a threshold number of candidates.

Fingerprint matching: Subsequently, among the candidates chosen in phoneme context matching, we consider a number of boolean features of each unit, e.g., whether the unit is stressed, word initial/final, phrase initial/final, sentence initial/final, etc. These features are represented as a bitset we call the *unit fingerprint*. This unit fingerprint F_u can then be scored against the context specific target fingerprint F_t as, $cost = popcnt(F_u \oplus F_t)$, where $popcnt$ returns the count of number of bits set to 1. This score can be computed quickly using compactly represented data, and it does not take any additional space in the index in our implementation. Using this score and the score from the context search, we then reduce the search set to 100 units which are the units used in the Viterbi search.

2.3. Model prediction

The classical unit selection approaches typically use either rule-based methods, hidden Markov models (HMMs) or neural networks (NN) for target cost [2, 8, 4, 5, 6, 7, 9], and acoustic distances for concatenation cost. This process usually involves many hand-tuning steps, and may not be flexbile when building multiple voices for multiple languages.

We propose to use a probabilistic approach to implement both target and concatenation costs to minimize manual-tuning. In particular, we employ deep and recurrent mixture density networks (MDNs) [10] to predict target and concatenation distributions, which are used to implement target and concatenation costs, respectively. As a combination of a conventional neural network with a probability distribution, MDN provides a framework to model conditional density functions [10, 11]. A deep feed-forward MDN uses linguistic features from the front-end at the current time step to predict acoustic and prosodic feature distributions through multiple layers of nonlinear transformations. A deep feed-forward MDN is formulated as

$$p(\mathbf{y}_t|\mathbf{x}_t;\lambda) = \Sigma_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{y}_t;\mu_k,\Sigma_k), \quad (1)$$

where $\mathbf{x}_t$ and $\mathbf{y}_t$ are the features for input and output at current time step, λ represents model parameters of a multiple-layer neural network, K is the number of Gaussian components, α_k, μ_k and Σ_k are the weight, mean and covariance of a Gaussian component.

However, feed-forward MDN does not have the ability to model temporal dependency, which is important for modeling speech, especially the fundamental frequency (f_0). To this end, we use recurrent mixture density network for modeling f_0. A recurrent mixture density network is formulated as

$$p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{x}_{t-1},\cdots,\mathbf{x}_1;\lambda) = \Sigma_{k=1}^{K}\alpha_k\mathcal{N}(\mathbf{y}_t;\mu_k,\Sigma_k^2). \quad (2)$$

Different from (1), $p(\mathbf{y}_t|\mathbf{x}_t,\mathbf{x}_{t-1},\cdots,\mathbf{x}_1;\lambda)$ depends on all the previous input features to predict the output distribution, and hence has the ability to model sequential data.

The input features to the neural nets are language- and context-dependent linguistic features from the front-end. In general, these features are: quinphones, stress, part-of-speech context, tone context (for tonal languages), prominence, sentence type and initial/final positional features for syllable, word, phrase and sentence; and also a positional feature for the half-phone to indicate first or second half. The output features consist of the duration of a half-phone, 13-dimensional Mel-frequency cepstral coefficients (MFCCs) as spectral features and fundamental frequency (f_0) at the beginning, middle (only for f_0), and end of a half-phone, and their deltas. In total, there are 58 features at the output layer. The features are listed in Table 1.

In practice, the feed-forward MDN (**MDN** in the remainder of the paper) is following the implementation in [11], but using rectified linear units (RELUs) at the hidden layers, while the recurrent MDN (**RMDN**) has two feed-forward hidden layers that use RELUs and uses a recurrent layer on top of the feed-forward layers. The recurrent layer employs gated recurrent units (GRU) [12], which is an alternative implementation of the long short-term memory (LSTM), but uses fewer model parameters to achieve similar performance, as validated in statistical parametric speech synthesis [13]. Both MDN and RMDN have the same input and output features. Although only parts of outputs from MDN and RMDN are used for cost calculation in a later step, all other features are still trained due to the benefit of multitask learning [14].

2.4. Cost calculation

The context-dependent input features to MDN and RMDN allow us to predict context-dependent variances, and they are

Table 1: *Output features of deep and recurrent neural networks for target and concatenation models.*

Feature	Half-phone position	Dimension	Model
Duration	-	1	MDN
MFCC-b	Beginning	13	MDN
MFCC-e	End	13	MDN
ΔMFCC-b	Beginning	13	MDN
ΔMFCC-e	End	13	MDN
f_0-b	Beginning	1	RMDN
f_0-m	Middle	1	RMDN
f_0-e	End	1	RMDN
Δf_0-b	Beginning	1	RMDN
Δf_0-e	End	1	RMDN
Total		58	

treated as context-dependent weights for both target and concatenation costs. The basis of the cost function is the negative log-likelihood where constant terms are eliminated. We formulate the cost function as

$$\mathcal{F}(\mathbf{y}|\lambda, \mathbf{w}) = \sum_k \alpha_k (\mathbf{w} \odot (\mathbf{y} - \mu_k))^T \Sigma_k^{-1} (\mathbf{y} - \mu_k) \quad (3)$$

where $\lambda = \{\alpha_k, \mu_k, \Sigma_k\}$ are mixture weight, mean and covariance predicted from the density network, $\mathbf{w}$ is a vector for predefined weights which are the same for all languages, and $\mathbf{y}$ is the output features of MDN and RMDN. In practice, we use single Gaussian and diagonal covariance.

We use the same cost function in Eq. 3 to implement both target and concatenation costs. The total cost is a combination of target and concatenation costs, defined as

$$\mathcal{C}(\mathbf{y}, \lambda, \mathbf{w}) = \gamma^t \mathcal{F}(\mathbf{y}^t|\lambda^t, \mathbf{w}^t) + \gamma^c \mathcal{F}(\mathbf{y}^c|\lambda^c, \mathbf{w}^c), \quad (4)$$

where γ^t and γ^c are predefined weights to balance target and concatenation costs, $\mathbf{y}^t$ and $\mathbf{y}^c$ are features used for target and concatenation, respectively, and λ^t and λ^c are corresponding model parameters for $\mathbf{y}^t$ and $\mathbf{y}^c$ features, respectively.

In practice, only f_0-m and duration are used as target features $\mathbf{y}^t$ for target cost, and ΔMFCC-e and Δf_0-e are used as concatenation features $\mathbf{y}^c$ to calculate a concatenation cost. The predefined weights $\mathbf{w}$, γ^t and γ^c are manually tuned based on human perception using en-US system, and are applied to all the other languages/voices.

2.5. Long units

Long units is a method of replacing a portion of a synthesized utterance with a block of pre-determined units that function as a single large unit [15]. Siri's long units were previously based on text matching with a set of text-pattern features. In the current system, our long units are based on phonetic matching, and use common features with the machine learning inputs and synthesis data pipeline. This leads to better decisions on when to use situational long units. In addition, we greatly reduced the number of long units used in the system, focusing them on extremely important sentences and non-speech sounds that are otherwise challenging for synthesis. While this technique is powerful, it has a high cost in terms of both voice data and recording time, so we utilize it in a targeted manner. In practice, only a small minority of our synthesized utterances now contain any predefined long units.

Table 2: *Memory hierarchy for a representative voice with one million units. Note that all the information presented in this table is stored on the device.*

Level	Total Size	Bytes / Unit	Units / Slice
Context Table	3.2M	3	1000
Fingerprints	4M	4	1000
Unit Index	75M	75	100
Audio Data	300M	320	1

The current system also features Siri's first instances of phonetically identical long units with substantial prosodic variation. In many cases, these variations could be described as having a certain sentiment or matching certain emotional state. Being able to choose between them assists the voice in sounding more human.

2.6. Optimizations

In order to make the system run on-device with low latency, we made optimizations to unit indexing and utilized parallelization.

Unit indexing plays an important role for both latency and footprint. Table 2 shows the memory footprint for the constituents of a representative voice. Our two-step preselection needs to refer only to the *Context Table* and *Fingerprints* sections of the voice, and only the former needs to be searched extensively in the phoneme context matching step. To simplify context lookup, all units in the database are arranged in an index, ordered by their quinphone context, in the order $\langle$phoneme, predecessor, successor, pre-predecessor, post-successor$\rangle$. This allows us to represent most matching subsets as a contiguous range of units in the index, and find the subsets with a simple binary search. Only the *Audio Data* for the ultimately chosen units is accessed in random order.

We achieve further performance improvements by parallelizing the calculations for model predictions, concatenation costs, and target costs, and by locally caching the 2,000 most commonly used utterances for the user.

3. Voice building

The Siri TTS project is conceived as a multilingual project. This means that we try to keep the voice-building process as language-neutral as possible. The process involves data recording, voice building, acoustic model training, and unit pruning.

3.1. Talent selection and data recording

Voice talent selection varies somewhat from a typical TTS voice selection process since there are some Siri-specific considerations. First and foremost, a voice must be perceived as being compatible with the Siri personality. This aspect is evaluated as part of the selection process. Second, Siri has male and female voices and they are developed as a pair. Recording scripts are designed for general-purpose synthesis coverage, but with an emphasis on likely domains of usage. Audio is recorded at 48 kHz, 16 bit. 10–20 hours of audio is recorded.

Discrepancies can exist between the text spoken by the voice talent and the script used during audio recording. To detect these discrepancies, we first build a language model that is interpolated with the script; then use this language model and a speaker-independent acoustic model to automatically transcribe the recordings; and finally we align the automatic transcription with the original script to detect discrepancies.

3.2. Voice building

Once the transcription of the audio is cleaned, we start the voice building that follows a two-step process: a) the audio is force aligned with the corrected transcription using a speaker-independent deep neural network acoustic model, b) a number of features, acoustic and symbolic, are added on per phone basis to the data, so that the voice consists of the recorded audio and an index into the audio that includes feature information. The audio is processed using a standard codec in order to reduce the overall voice size.

3.3. Pruning

In order to reduce the footprint of voice/audio data, we prune unused and rarely used units. In our experiments, we pruned nearly half of the units, which means we reduced the footprint of voice data by nearly half.

We perform the following three steps to prune units [16]. First, we generate speech using the engine presented in this paper with unpruned voice data over a large corpus. The corpus we used contains Siri logs from 2015 with more than 200 million words, since our main focus here is to improve the naturalness of Siri speech.

Then, we calculate the statistics of unit usage. The internal results of the statistics show that about 41% of the units are never used. Furthermore, about 8.3% of the units are used less than 10,000 times in this large usage corpus, and about 8.8% of the units are used more than million times. Note that a unit that is never used does not mean it never appears in the Viterbi search.

Finally, we prune out unused and rarely used (i.e., used less than 10 times) units based on the statistics. Initially, we expected that our TTS engine with the pruned voice should produce the exact same output as the unpruned voice, however this is not the case, even if we only prune unused units. We found that when pruned units are replaced by other units during preselection, they become unit candidates in the Viterbi search algorithm. This sometimes produces a better Viterbi path, such that a different output is synthesized.

With unit pruning, the sizes of unit index and audio data presented in Table 2 are significantly reduced. A subjective preference test confirmed that unit pruning does not degrade the naturalness (no statistically significant difference).

4. Experiments

We compared the baseline system deployed in iOS 9 with the system described in this paper (named as **NN-system** below). The baseline is a classical unit selection system and using corpora with 22 kHz sampling rate, while the new system is using 48 kHz data. Note that both systems use the same front-end for linguistic feature extraction and only the back-end (including preselection, model prediction, and cost calculation) is different.

We first performed two ABX preference tests using a female Mandarin Chinese voice to show the impact of sampling rate, assuming the impact of sampling rate is language- and speaker-independent. Ten native Mandarin Chinese speakers participated in both tests, in which each listened 30 randomly selected pairs of utterances and were asked to choose the one in each pair that sounded more natural to them or choose the "neutral" option if no preference. The utterances within each pair came from differing systems but had the same linguistic content. The results are presented in Table 3. It is observed that with the same sampling rate, the NN-system is significantly better than the baseline system. It implies that all the improvements we made in the NN-system are leading to better quality of synthetic speech. The experimental results also show that increasing sampling rate from 22 kHz to 48 kHz can lead to significant improvement in naturalness, and this is consistent with reported work on statistical parametric speech synthesis [17].

Table 3: *Preference test results for the impact of sampling rate using a female Mandarin Chinese voice. p-values are smaller than 0.01 for both tests.*

Baseline	NN-system (22k)	NN-system (48k)	Neutral
11.3%	**73.6%**	-	15.1%
-	21.6%	**27.3%**	51.1%

Table 4: *Mean opinion score (MOS) results on female voices of seven languages. All the differences between the baseline and NN-systems are statistically significant.*

Language	Baseline	NN-system
en-US	2.95 ($\pm$ 0.10)	3.73 ($\pm$ 0.09)
en-GB	2.65 ($\pm$ 0.10)	3.45 ($\pm$ 0.11)
en-AU	2.07 ($\pm$ 0.11)	3.90 ($\pm$ 0.10)
es-ES	2.30 ($\pm$ 0.11)	3.67 ($\pm$ 0.11)
it-IT	3.06 ($\pm$ 0.12)	3.59 ($\pm$ 0.11)
ru-RU	2.82 ($\pm$ 0.12)	3.76 ($\pm$ 0.12)
zh-CN	2.69 ($\pm$ 0.11)	3.51 ($\pm$ 0.09)

We then subjectively evaluated each system entirely, NN-system (48 kHz) against the baseline (22 kHz), and did this across multiple languages. For each language, we asked 30 native speakers to give judgements based on a 5-point discrete scale Mean Opinion Scores (MOSs) labeled "Bad", "Poor", "Fair", "Good", and "Excellent". Each listener listened to 30 sentences synthesized from two different systems, for a total of 60 stimuli. The evaluation was conducted on listeners' own iPhones using an internal iOS app and the listeners can use any headphone as they like to simulate real user conditions.

The MOS results are presented in Table 4. It is clear that the NN-system is found to be significantly better than the baseline system in all cases. As we mentioned earlier, the front-end is identical for the baseline and NN-systems, hence the improvements are mainly from the improvements we made in the back-end, i.e., pre-selection, deep learning guiding, and long units.

5. Conclusions

In this paper, we described Apple Siri's deep learning-guided unit selection text-to-speech synthesis system, which provides the voices for Siri and Apple Maps. The system uses deep and recurrent mixture density networks to implement target and concatenation costs. We presented multiple optimizations of the run-time engine for low latency, low footprint, and high quality synthetic speech, and the multilingual voice building process. The TTS system has been deployed into hundreds of millions of desktop and mobile devices in multiple languages, and has significantly improved user experience.

6. Acknowledgements

We would like to thank our colleagues Alex Acero, Woojay Jeon, and Bunny Laden for useful comments and proofreading.

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

[3] S. P. Kishore and A. W. Black, "Unit size in unit selection speech synthesis." in *Interspeech*, 2003.

[4] Z.-H. Ling, L. Qin, H. Lu, Y. Gao, L.-R. Dai, R.-H. Wang, Y. Jiang, Z.-W. Zhao, J.-H. Yang, J. Chen *et al.*, "The USTC and iflytek speech synthesis systems for Blizzard Challenge 2007," in *Blizzard Challenge Workshop*, 2007.

[5] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "Using deep bidirectional recurrent neural networks for prosodic-target prediction in a unit-selection text-to-speech system." in *Interspeech*, 2015, pp. 1606–1610.

[6] T. Merritt, R. A. Clark, Z. Wu, J. Yamagishi, and S. King, "Deep neural network-guided unit selection synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5145–5149.

[7] L.-H. Chen, Y. Jiang, M. Zhou, Z.-H. Ling, and L.-R. Dai, "The USTC system for Blizzard Challenge 2016," in *Blizzard Challenge Workshop*, 2016.

[8] 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.

[9] X. Gonzalvo, S. Tazari, C.-a. Chan, M. Becker, A. Gutkin, and H. Silen, "Recent advances in google real-time HMM-driven unit selection synthesizer," in *Interspeech*, 2016, pp. 2238–2242.

[10] C. Bishop, "Mixture density networks," *Tech. Rep. NCRG/94/004, Neural Computing Research Group, Aston University*, 1994.

[11] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014, pp. 3872–3876.

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[13] Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5140–5144.

[14] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4460–4464.

[15] N. Campbell, "Conversational speech synthesis and the need for some laughter," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1171–1178, 2006.

[16] H. Lu, W. Zhang, X. Shao, Q. Zhou, W. Lei, H. Zhou, and A. Breen, "Pruning redundant synthesis units based on static and delta unit appearance frequency," in *Interspeech*, 2015.

[17] A. Stan, J. Yamagishi, S. King, and M. Aylett, "The Romanian speech synthesis (RSS) corpus: Building a high quality HMM-based speech synthesis system using a high sampling rate," *Speech Communication*, vol. 53, no. 3, pp. 442–450, 2011.