uRobo: An End-to-End Concatenative Speech Synthesis System

Tyrus Cukavac

December 21, 2017

Abstract

This paper introduces uRobo, an open concatenative speech synthesis system built using the Kaldi framework for its Automated Speech Recognition layer and utilizing deep recurrent neural networks for target feature predictions in the unit selection stage. uRobo is an end-to-end system, meant to automate construction of a final model from raw data to synthesized speech. Initial target feature prediction models from uRobo were trained on ten hours of the LibriSpeech audio corpus comprising female voices, while units used for concatenation were tested with 10 hours of heterogenous speaker data, as well as roughly 25 minutes of data from a single speaker. Moreover, uRobo is capable of unit selection at the triphone level (with diphone and monophone backoff) as well as the monophone level. We show that uRobo is able to produce intelligible voices with both models, but the triphone-diphone-monophone backoff architecture produces more intelligible and slightly smoother results.

1 Introduction

The generation of artificial, yet natural-sounding speech is one of the oldest problems in computer science. Although production of a general, intelligible synthetic voice is a worthy goal, there also exists a need to create voices which sound like a target speaker. This need is particularly acute in the domains of healthcare and media production. Many people who must have their voiceboxes removed for medical reasons such as cancer, often wish for the ability to re-create their orginal voice. Moreover, for film production companies, imperfect audio captured during filming force them to engage in time-intense and expensive audio recording sessions with their talent in order to meet production deadlines.

1.1 Existing Architectures

State of the art text-to-speech architectures tend to fall into one of three categories: a parametric approach, an end-to-end neural network approach, and a concatenative approach.

The parametric architecture essentially works to make predictions on durations of phonemes as well as the acoustic features necessary for synthesis. These acoustic features are then provided to a vocoder which synthesizes the sounds into waveforms.TKTKThe parametric model described by Zen, et. al. utilizes an LSTM (Long Short Term Memory) RNN (Recurrent Neural Network) to make predictions of the above features to be provided to the vocoder.[6] Although parametric models are resilient in the sense that they are capable of producing features for any given frame (in comparison to other models that may be limited by the unit data collected during trainign) the results are often apparently from a machine and lack a natural sound.

Parametric architectures need not be the only synthesis approach using neural networks. The most recent state of the art architectures for text-to-speech synthesis make use of neural networks to generate pure waveforms from input text. DeepMind's Wavenet uses dilated causal convolutional neural networks[1], and Baidu's Deep Voice passes inputs through layers of recurrent neural networks. Both of these approaches result in synthetic voices that human listeners seem to find more natural than previous approaches[5], but require a great deal of computational power and time to train. Moreover, this area of study is still in relative infancy as opposed to other approaches, such as the concatenative, which have had more time to mature.

The concatenative approach, described by Hunt and Black, at its most basic level selects pre-existing units of audio from a database of utterances, stitching these together and performing basic post-processing to create a final audio output[3]. Obviously to provide adequate coverage for a given language and a given speaker, a great deal of audio data must be collected. This unfortunately leads to additional problems, such as slow database lookups for large candidate units for a given frame/phone. Google's advances in their own concatenative model have helped them address several of these issues by optimizing search for ideal unit candidates and extending the size of the units used to build utterances.[2]

Despite the above problems, the concatenative approach has yielded more natural-sounding results in many cases than the parametric approach, due to its use of existing waveforms rather than the synthesis of new waveforms by a vocoder, which often results in artifacts. For the purposes of imitating a given speaker, the concatenative approach may seemingly yield the best of all worlds: taking less time and resources than an end-to-end neural network while also being able to match actual phonetic level sounds of a given speaker. As a result, the uRobo system follows this architecture to synthesize speech.

2 The Concatenative Approach

The concatenative approach to synthesis is at its heart a dynamic programming problem. Given a database of units which are in essence waveforms with acoustic features, the algorithm attempts to find a path/sequence of units corresponding to a given sequence of words/phonemes resulting in the minimum total cost value over all units.

$\overline{u}_1^n = min_{u_1,\ldots,u_n} C(t_1^n, u_1^n)$

wherein $\overline{u}_1^n$ is the final sequence of n units required for a new utterance that are selected by the algorithm. $u_x, x \in n$ is a given unit at index x and $t_x, x \in n$ is a target "unit" or rather a representation of certain features that a unit at index x should have. C is a cost function over a sequence of units.

The cost value per unit can be defined as a function of two separate costs. The target cost C^t, which is essentially a weighted sum of absolute differences between a target feature vector and a candidate unit's feature vector, as well as a concatenation cost C^c, which is the weighted sum of absolute differences between a candidate unit and a previous candidate.

$C^t(t_i, u_i) \sum_{j=1}^{p} w_j^t |t_{i_j} - u_{i_j}|$

$C^c(u_{i-1}, u_i) \sum_{j=1}^{q} w_j^c |u_{i-1_j} - u_{i_j}|$

Where p represents the number of features in the target feature vector and q represents the number of features in the concatenation feature vectors of the two units. The cost for a given unit, then, is the sum of $C^c and C^t$. We can thus expand the cost of a sequence to be

$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=1}^{n} C^c(u_{i-1}, u_i)$ [3]

Obviously calculating this for every possible path would be an intractable problem, therefore the Viterbi algorithm is used to find the minimum cost sequence. The problem can be visualized as a sparse matrix wherein each column corresponds to the set of candidate costs for a given unit. The first column of the matrix is initialized as the concatenation cost between the first unit and silence. In the remaining iterations, unit by unit, each candidate is compared against the 1)target feature vector and 2) all previous units. The minimum concatenation cost is selected as the final concatenation cost for that unit, and the previous unit is stored in a backtracking matrix. Retrieving the final sequence of units is simple using a backtracing algorithm. This final sequence of units i.e. selected waveforms, are then stitched together to create the final waveform representing an utterance.

```
1:  function VITERBI-UNIT-COSTS(t_1^n, u_1^n)
2:      matrix cost
3:      matrix back
4:      for each u_{1_x} in u_1 do
5:          cost_{1,x} = C^c(SIL, u_1)
6:      for each u_i in u_2^n do
7:          for each u_{i_x} in u_i do
8:              C^t(t_i, u_{i_x})
9:              C^c(u_{i_x}) = min C^c(u_{i-1}, u_{ix})
10:             b = argmin C^c(u_{i-1}, u_{ix})
11:             cost_{i,x} = C^t(t_i, u_{i_x}) + C^c(u_{i_x} + cost_{i-1,b}
12:             back_{i,x} = b
        return cost, back
```

```
1:  function BACKTRACE-UNITS(cost, back)
2:      b = argmin cost_n
3:      u_n = b
4:      for each i ∈ (n − 1) → 1) do
5:          b = back_{i+1,b}
6:          u_i = b
        return u_1^n
```

3 uRobo Architecture

The uRobo system is composed of three separate layers which combine to produce a synthesis model. First, the system needs to be able to decompose audio in a manner that features can be extracted for use in unit selection, and therefore requires a speech recognition system in order to produce data usable by the concatenative architecture. Next, a system is required to preprocess the raw output of the ASR in order to extract additional features and put them into a form that the neural network can learn from. Finally, a unit selection system using the viterbi algorithm and the extracted data can be used to acquire units and ultimately synthesize an audio file.

3.1 Speech Recognition System

For automated speech recognition, the Kaldi ASR framework was utilized. Kaldi has pre-built scripts that allow a user to train an ASR system on the LibriSpeech corpus[4], an open corpus that is the foundation of the data used by the concatenation synthesis system. Kaldi's training script for ASR downloads the LibriSpeech corpus as well as the language model files generated from an analysis of that corpus, i.e. vocabulary, phones, and a lexicon translating words to various phone sequences.

Kaldi then goes through all of the data, a total of 1000 hours, and builds triphone based alignment models utilizing HMM-GMMs and decision trees, extracting features such as MFCCs as well as performing multiple alignments on the data to improve each successive model.

with a final alignment model completed, the uRobo system can call kaldi scripts to perform forced alignment on the librispeech data. For the purposes of development and the experiments to follow, the clean training data of Librispeech, comprising 100 hours of audio data, as well as the clean test data, were aligned and output into a text format that specified a given utterance's timecode to specific phones.

Of particular use

3.2 Preprocessing

3.3 Unit Selection Using a Deep Recurrent Neural Network

Hunt and Black present the general concept behind the Concatenative approach, but there is still the question of how to generate target features for computation of the target cost C^t. Much work h

4 Experiment Setup

4.1 The Corpus

Much of the difficulty in this area lies in a lack of substantial and usable training data. Many of the corpora used by large corporations incorporate many hours of audio data recorded by professional voice actors, and are closed to the public. The open Librispeech corpus[4] aims to fill part of that gap by formatting and segmenting data from the LibriVox project, resulting in thousands of hours of audio data. Unfortunately, each speaker tends to only contribute around 30 minutes of audio to the overall project, meaning any attempts at making use of this data for a text-to-speech model will be dealing with varied voices.

4.2 Utterance Synthesis and Metrics

5 Results

6 Conclusions and Avenues for Future Work

Test

References

[1] Sercan Ömer Arik, Mike Chrzanowski, Adam Coates, Greg Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep voice: Real-time neural text-to-speech. *CoRR*, abs/1702.07825, 2017.

[2] Xavi Gonzalvo, Siamak Tazari, Chun an Chan, Markus Becker, Alexander Gutkin, and Hanna Silen, editors. *Recent Advances in Google Real-time HMM-driven Unit Selection Synthesizer*, Sep 8-12, San Francisco, USA, 2016.

[3] A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1996. On Conference Proceedings., 1996 IEEE International Conference - Volume 01*, ICASSP '96, pages 373–376, Washington, DC, USA, 1996. IEEE Computer Society.

[4] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. pages 5206–5210, 04 2015.

[5] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[6] Heiga Zen, Yannis Agiomyrgiannakis, Niels Egberts, Fergus Henderson, and Przemyslaw Szczepaniak. Fast, compact, and high quality LSTM-RNN based statistical parametric speech synthesizers for mobile devices. *CoRR*, abs/1606.06061, 2016.