NEURAL SPEECH-TO-SPEECH SYNTHESIS

Jared Samet - UNI: jss2272

jss2272@columbia.edu

ABSTRACT

Prosody features that transfer across speakers can be extracted using unsupervised learning. By adding the cluster labels as annotation to text, a text-to-speech synthesis system can learn to incorporate the prosody into its output. The resulting system can produce audio that is different even when the text is the same and the resulting audio can mimic the prosody from the original speaker. The abstract should be about 175 words total.

Index Terms— Prosody, unsupervised learning, speech synthesis, seq2seq

1. INTRODUCTION

I used Kaldi [1] to create an alignment for the Tedlium data using the final triphone model it created. For each vowel phoneme, I used Kaldi's pitch extractor and the first MFCC component (energy) to create two series of numbers. Kaldi's pitch extractor is already normalized but I used [StandardScaler] to normalize the energy component across the utterance. I fit a second-degree (?) Legendre polynomial to the pitch and power to create six features for each vowel. The duration gave me the seventh feature. I then ran K-means clustering on these to create eight different vowel clusters that were common across the entire range of speakers in the (subsampled) Tedlium data.

I then used the same triphone model to generate an alignment for the LJSpeech dataset, extracted the same pitch, power, and duration features for LJ, and used the previously computed vowel clusters to assign a cluster label to each vowel in the LJSpeech dataset. I (slightly) modified the Tacotron implementation to accept a sequence of tokens instead of a sequence of characters. Instead of text characters, my input tokens consisted of (Kaldi's) phonemes and the vowel cluster labels. Having suitably modified Tacotron, I then trained Tacotron on the [phoneme + cluster label, audio] pairs.

Finally, to see if it worked, I recorded myself saying a sentence in multiple ways, ran each .wav through the same align + label steps, and fed the resulting [phoneme+label, text] pairs to Tacotron. She said the same thing different ways.

2. RELATED WORK

This project involved two main components: first, extracting prosody features from a set of input audio files; and second, training a text-to-speech synthesis model on a dataset that had been labeled using the extracted prosody features.

Selkirk [2] discusses sentence prosody and pitch accent in the context of English. Although English is generally not thought of as a tonal language, Selkirk writes that "[i]n English a pitch accent associates to a stress-prominent syllable in a word (typically the main word stress.)" Ghahremani et al. [3] describes an pitch-extraction algorithm ("the Kaldi pitch tracker") based on Talkin [4] that is specifically designed for use in the speech recognition concept and is implemented in the open-source Kaldi project []. This project uses that implementation to extract the pitch contour. Fujisaki [5] models the F_0 contour over the duration of an utterance as the sum of a set of impulse response and step response functions, parameterized with a finite number of scalar values. Wang et al. [6] use the pitch and amplitude contours to improve tone recognition in Mandarin by identifying "maxima, minima, and inflection points of particular acoustic events." Wong and Siu [7] use robust regression and orthogonal polynomials to create features for a decision tree classifier in order to recognize tones in Chinese lan guages. Finally, Lin [8] and Mary [9] use a small number of Legendre polynomial coefficients to represent the pitch contour as a finite-dimensional feature vector, which is the approach used in this project.

Speech synthesis or text-to-speech is a well-studied problem that has been actively researched since the 1950s. While there has been remarkable progress in the field in recent years, the quality of computer-generated speech has not yet reached human levels. Current commercial systems described in Khan et al. [10] and Taylor [11] generally use concatenative speech synthesis to produce their output. However, the alternative approach of parametric synthesis using neural networks is rapidly gaining popularity, with several papers since 2016 demonstrating impressive results in the quality of the output. The first of this generation was Google's WaveNet (Oord et al. [12]), followed in quick succession by Deep Voice and Deep Voice 2 from Baidu (Arik et al. [13], [14]), Char2Wav from MILA (Sotelo et al. [15]), and Tacotron from Google (Wang et al. [16]). Each of these systems has taken a different approach to the network architecture to address different

aspects of the speech synthesis pipeline. Tacotron, which is the backend used in this project, is a nearly end-to-end text-to-speech system based on the sequence-to-sequence with attention model. As the authors describe, "The model takes characters as input and outputs the corresponding raw spectrogram, which is then fed to the Griffin-Lim reconstruction algorithm to synthesize speech."

3. OVERVIEW

The goal of this project was to create a system that, given an input audio file from an arbitrary speaker, produces a synthesized audio output of the same utterance in the voice of a second speaker, where the prosody of the output audio matches that of the input audio as closely as possible. The system implemented uses a pipeline of several processing steps in order to accomplish this. An overview of the pipeline and a diagram are presented here for context; a detailed description of each step follows.

The first portion of the system is the vowel-cluster training process, which takes as input a previously-trained acoustic model for alignment and a speech dataset from multiple speakers. As output, it produces a clustering model that can be used to annotate an audio utterance from an arbitrary speaker with cluster labels for each vowel in the utterance. This portion of the system uses Kaldi to, first, perform forced alignment on the multi-speaker dataset, and, second, to extract the pitch contour and the first (energy) MFCC component for each frame of the input audio. Given the alignment, pitch, and power contours, an unsupervised clustering algorithm (K-means) trained on the audio segments corresponding to vowels to learn several distinct ways in which syllables can be pronounced.

The second portion of the system is the single-speaker annotation process, which takes as input the pre-existing acoustic model, the newly-trained vowel-cluster model, and a large speech dataset from a single speaker. As output, it produces a trained Tacotron model that can be used to generate synthesized utterances. This portion of the system first uses Kaldi to perform forced alignment on the speech dataset and extract the pitch and power features, as before. It then uses the vowel-cluster model to produce an annotated phoneme sequence for each utterance in the single-speaker dataset. Finally, the audio and the annotated phoneme sequence pairs are used to train the Tacotron model.

The final portion of the system is generates new utterances. As input, it takes the pre-existing acoustic model, the newly-trained vowel-cluster model, and the newly-trained Tacotron model, and an input audio file in the voice of an arbitrary speaker. As output, it produces synthesized audio of the equivalent utterance where the prosody matches that of the input utterance as closely as possible. This portion of the system computes an alignment for the input utterance and extracts the pitch and power features; uses the vowel cluster model to produce an annotated phoneme sequence for the input utterance; and, finally, uses the newly-trained Tacotron model to synthesize the output audio.

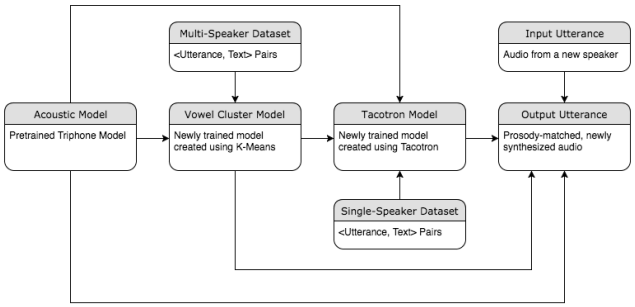

Fig. 1. Pipeline Overview

4. PROSODY FEATURE EXTRACTION

All three portions of the pipeline involve extracting prosody features from the input audio – in the vowel-cluster training process, the input audio is the multi-speaker dataset; in the single-speaker annotation process, the input audio is the single-speaker dataset; and in the utterance-generation portion, the input audio is the new utterance the user wishes to re-synthesize in a new voice. The prosody feature extraction is performed in three steps.

First, Kaldi's `align.sh` script uses a pretrained acoustic triphone model – in this project, the final triphone model resulting from Kaldi's TEDLIUM recipe – to compute a forced alignment of the input audio, and Kaldi's `ali-to-phones` tool is used to convert the model-level alignment to a sequence of *(phone_id, start_time, end_time)* tuples. Next, Kaldi's `make_mfcc_pitch.sh` script creates the MFCC and pitch features for each frame of the input audio, the `copy-matrix` tool converts this to a text file, and my python script `kaldi_to_npz.py` converts the text file to a numpy array (.npz) file. Finally, seven real-valued features are created for each vowel phone in the alignment.

The first three features are the coefficients of the second-degree Legendre series that is the least-squares fit to a series of (x, y) points where the length of the series is the computed number of frames plus 4, the x values are evenly spaced between -1 and 1, and the y values are the frame-by-frame pitch values computed using Kaldi's pitch-extraction algorithm, starting two frames before the beginning of the phone and ending two frames after the end of the phone. Since Kaldi's algorithm already normalizes the pitch contour over a three-second window, no further normalization is done before computing the Legendre coefficients.

The next three features are the Legendre coefficients for the power (first MFCC component) component, which is normalized to have mean zero and variance one over the whole

utterance before computing these coefficients. The coefficients are calculated in the same way as for the pitch features. The seventh feature is simply the duration of the phone.

These seven features were chosen in the hopes of maximizing the useful prosodic information available in a small set of numbers per phoneme. The two frames (20 ms) before and after the utterance are added to the sequence to improve the conditioning of the least-squares fit matrix; to compensate for potential slight errors in the alignment as computed by Kaldi; and to provide a small degree of context for the vowel in question. The domain is fixed at [-1, 1] so that the coefficients capture the level, slope and curvature over the length of the phoneme, regardless of the duration. Due to the structure of the Legendre polynomials, the three features for the pitch and power series contain information about whether the vowel is pronounced with a rising, falling, or flat tone, and whether the vowel contained a local maximum or minimum of pitch or power. Finally, the duration feature is a simple attempt to detect whether the vowel is pronounced quickly or is drawn-out.

Show what the Legendre coefficients look like for some different curves. Show what the actual pitch and power curves look like for some brief, manually labeled utterances. Describe the Kaldi pitch extractor. Definitely include the Kaldi paper as a reference. Explain how the Legendre coefficients work. Explain why these were a sensible way of capturing prosody. Talk about pitch envelopes and tonal languages.

5. CLUSTERING VOWELS

Unsupervised learning FTW. Discuss why we should expect there to be clusters even in an atonal language like English. Talk about stressed vs unstressed as initial motivation but how there are probably more things like this. Talk about how stress is mostly a pitch change.

I ran K-means clustering on my seven features and created eight vowel clusters. I'm pretty sure I ran StandardScaler on the features first so K-means didn't get confused, double check this. This was intended to be enough to capture the major variation across how different vowels can be pronounced but not so many as to result in too-few training examples. I originally did this for each vowel separately, and only for a single speaker, but then I decided that was stupid so I did it across all speakers and across eight vowels. So there are only eight clusters. Here I need to demonstrate that the clusters are in fact "semantically" different in some way. Maybe include some metric of these or run TSNE on the coefficients.

I could probably have also just used the actual Legendre coefficients themselves but this would have required tinkering with the Tacotron internals more to accept continuous-valued features as part of the sequence instead of just a one-hot encoded value. This is something that could go in a future work section.

6. TACOTRON

Describe the Tacotron architecture and explain why it was easy to add the cluster labels.

6.1. Subheadings

Just for reminder.

6.1.1. Sub-subheadings

Just for reminder.

7. RESULTS

Find some way to quantify that it actually did something beyond "she never stole my money".

Try and quantify that the different clusters are actually different in some way. This is probably the most important section. Quantify if they are different from male to female speakers in any way.

Try and quantify that the speech result is better for my Tacotron than for without annotations. Say why this could be useful even if no one wants to do speech to speech.

Try and quantify that the output is actually preserving stuff from the original speech dataset.

8. DISCUSSION

9. LIMITATIONS

10. FUTURE WORK

11. ACKNOWLEDGEMENTS

I would like to thank Keith Ito for his outstanding open-source implementation of Tacotron. This project would not have been possible without his work. I would also like to thank Dan Povey, the lead developer of Kaldi, which was also essential to this project. Finally, I would like to thank Professor Beigi for teaching this class, which has been a pleasure!

(a) Result 1

(b) Results 3 (c) Result 4

Fig. 2. Example of placing a figure with experimental results.

12. REFERENCES

[1] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.

[2] Elisabeth Selkirk, "Sentence prosody: Intonation, stress, and phrasing," *The handbook of phonological theory*, vol. 1, pp. 550–569, 1995.

[3] Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 2494–2498.

[4] David Talkin, "A robust algorithm for pitch tracking (rapt)," *Speech coding and synthesis*, vol. 495, pp. 518, 1995.

[5] Hiroya Fujisaki, "Information, prosody, and modeling-with emphasis on tonal features of speech," in *Speech Prosody 2004, International Conference*, 2004.

[6] Siwei Wang and Gina-Anne Levow, "Mandarin chinese tone nucleus detection with landmarks," in *Ninth Annual Conference of the International Speech Communication Association*, 2008.

[7] Pui-Fung Wong and Man-Hung Siu, "Decision tree based tone modeling for chinese speech recognition," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. IEEE, 2004, vol. 1, pp. I–905.

[8] Chi-Yueh Lin and Hsiao-Chuan Wang, "Language identification using pitch contour information," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*. IEEE, 2005, vol. 1, pp. I–601.

[9] Leena Mary, *Extraction and representation of prosody for speaker, speech and language recognition*, Springer Science & Business Media, 2011.

[10] Rubeena A Khan and JS Chitode, "Concatenative speech synthesis: A review," *International Journal of Computer Applications*, vol. 136, no. 3, pp. 6, 2016.

[11] Paul Taylor, *Text-to-speech synthesis*, Cambridge university press, 2009.

[12] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[13] Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Jonathan Raiman, Shubho Sengupta, et al., "Deep voice: Real-time neural text-to-speech," *arXiv preprint arXiv:1702.07825*, 2017.

[14] Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," *arXiv preprint arXiv:1705.08947*, 2017.

[15] Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio, "Char2wav: End-to-end speech synthesis," 2017.

[16] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al., "Tacotron: A fully end-to-end text-to-speech synthesis model," *arXiv preprint arXiv:1703.10135*, 2017.