# Voice Translator (Project Report B)

**group:** ian calloway, heming han, seth raker, samuel tenka

**date:** 2016-11-20

**descr:** Report B for Eecs 351 Project, to be published on our website and reviewed by other teams. This document has the following outline:

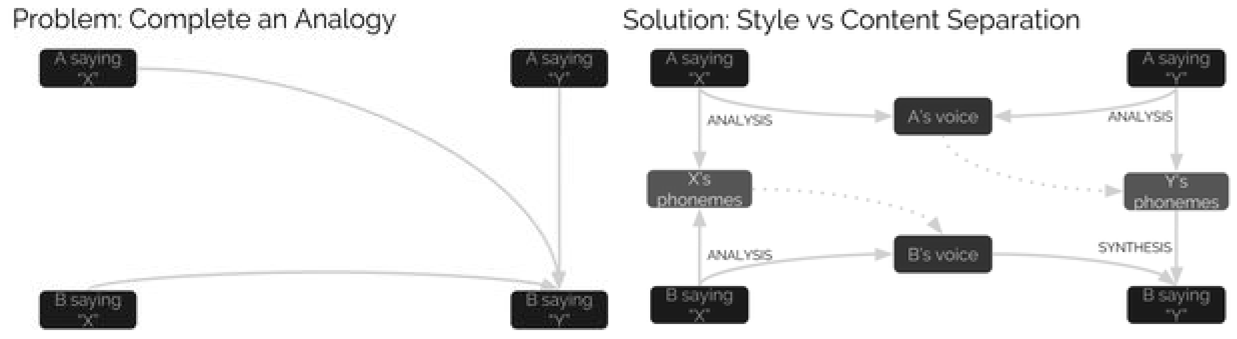
## 0. Introduction

*For last year's words belong to last year's language*
*And next year's words await another voice.*
--- T.S. Eliot

### *0.0. Problem and Applications*

The Voice Translator team aims to imitate human speech: if we listen a lot both to Queen and Birdy, we become familiar with the nuances of each artist's sound. We can imagine Birdy covering a Queen song, thereby mixing her voice and style with Queen's lyrics or content. The human brain can thus solve the analogy: "*What is to Birdy as `Don't Stop Me Now` is to Freddie Mercury?*" The question is, can a computer solve the same analogy?

We restrict ourself to plain human speech for now, but expect that our methods will generalize to domains such as music and general audio. Precisely, then, we seek to solve the ***voice morphing*** problem, which we view as a problem in analogy completion, and will solve via style/content separation:



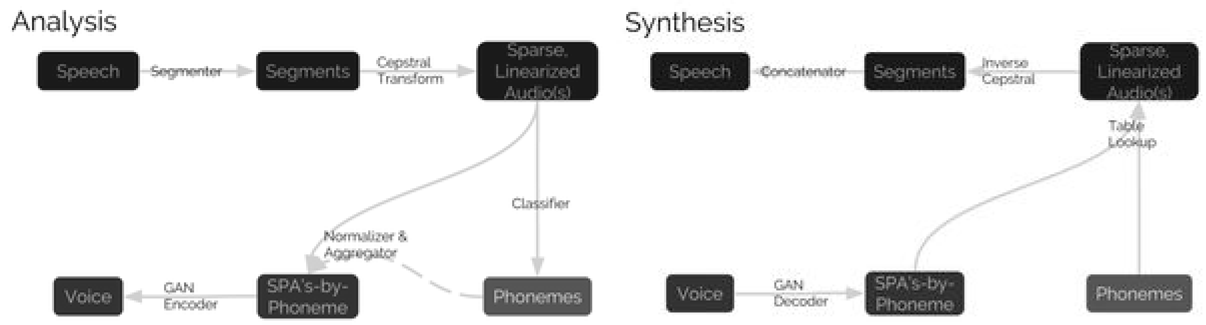
Before diving into our project details, we step back and ask: *What good is all this?* Well, the project excites us in multiple ways: we see it as a chance to explore modern techniques in handling rich data, and it also has exciting applications:

0. Music-creation
1. Synthesis systems for those without a voice
2. Emotion-detection based on voice (e.g. in a gps app to detect road rage)
3. Art beyond the creation of melodic music
4. Pranks of false identity

Finally, we just think it's really cool.

## *0.1. Architecture*

Key to our approach is the ***analysis*** and ***synthesis*** of style and content. In the domain of natural voices, ***style*** means a representation of voice qualities (accent, tempo, scratchiness, pitch, nasality, and so forth) independent from the phonetic ***content*** of audio produced in that voice. We break down the analysis task into ***segmentation***, ***classification***, and ***encoding***, and synthesis, correspondingly, into ***concatenation***, ***lookup***, and ***decoding***:



The segmenter will split given audio (~10 seconds) into small chunks (~0.2 seconds), each corresponding to a single phoneme. We identify that corresponding phoneme by transforming to a set of expert-crafted audio features, then applying a standard classifier. This produces the phonetic transcription part of the analyzer output. Aggregating the audio chunks belonging to a given phoneme, then compressing (encoding) those chunks into a small summary, we arrive at a representation of voice qualities. We perform synthesis via the same methods, inverted and in reverse order.

In sum, we have decomposed the voice morphing problem into pairs of inverse subproblems: segmentation / concatenation, classification / lookup, and encoding / decoding. For algorithmic details, see Section 4.

## ***0.2. Potential Obstacles***

Our project goal is the development of a certain function from a rich dataspace to a rich dataspace. For us, data is ***rich*** when it is represented by many raw low-level features (amplitudes at each audio sample), and that these raw features are related to the high-level features of interest (voice quality, phonetic transcription, etc) by no direct means. Because the system output in particular is rich, learning and validation pose problems: there is no obvious psychologically meaningful distance function on audio files. So, as discussed in Section 3, we'll validate the system piece-by-piece, and use heuristics for end-to-end validation. For training, we will explore recent unsupervised metric-learning techniques such as Generative Adversarial Networks.

A related challenge lies in the discovery of a task-appropriate high-level intermediate representation of voice quality. The number and nature of voice qualities is, to us, currently unknown. It will be crucial that we review the speech-processing literature. Black-box methods robust to domain, such as neural nets, may overcome this problem.

Finally, since phonemes interact, concatenation is nontrivial. As a reach goal, we'll train our model to contextualize phonemes given their neighbors, and, likewise, use context to better filter out non-phoneme noise.

## 1. Prior Work and Tools

*If I have seen further, it is by standing on*
*the shoulders of giants.*
--- I. Newton

*Einstein was a giant. His head was in the clouds,*
*but his feet were on the ground.*
*But those of us who are not*
*that tall have to choose!*
--- R.P. Feynman

### *1.0. Pre-Existing Software Tools*

We narrowed down our previous list of technologies. Listed from input-side to output-side of our data pipeline:

0. **Audacity**: Record voice samples, preprocess, and create spectrograms.

1. **audioop**: Raw audio manipulation in Python. Multitude of useful functions, including audioop.avg(fragment, width) to average samples for a given width and audioop.findfit(fragment, reference) to match a reference to a portion of a fragment.

2. **Numpy**: The fundamental Python package for scientific computing. Our linear algebra workhorse for comparing high-level voice features
.
3. **Keras**: A high-level neural-network library in Python built on Tensor Flow. Tensor Flow was originally developed by Google's machine intelligence research team to streamline development of networks for such problems as image classification and machine translation. Neural nets may provide the requisite degrees of freedom to model realistic voices.

4. **Matlab**: The language of our translator and design-phase analysis. Industrial-strength matrix algorithms (applicable, for instance, to quantified voice qualities) will form a base for our final system. The Audio System Toolbox provides a variety of functions, including filtering and equalization.

## *1.1. Voice Morphing: Reflections on Prior Work*

Popa et al briefly survey work (up to 2012) on the voice morphing problem, suggesting that all prior approaches suffer from (A) artifacts due to unnatural windowing in the time domain and (B) “over-smoothing” of synthesized features.

Such over-smoothing we are familiar with in the context of least-squares models: if such a model is uncertain, it will output an average of its perceived possibilities. While such models work well when reality has a convex loss function, they perform poorly for more complex situations or more raw feature representations. For instance, is what is “the” average of a baby and a senior citizen? Linear models on the most accessible features would answer: “a white-haired human that crawls with the aid of a cane, and has completed half of high school”, while a model based on deeper features might answer the more natural “a young adult”. We are struck at the precision of the human ear: when realism and variety are our goals, it seems an even more challenging task to synthesize audio than images. Plain linear models are inadequate, and Popa et al’s survey confirms this.

An initial step toward nonlinearity is to have linear weights adapt to the input; another is to condition successive algorithm steps on a classifier’s output. Those two ideas form the main new development of the Popa paper; they call their method “Local Linear Transformation”. However, their method improves only modestly on prior work on a subjective, relative scale, and they do not report any absolute evaluation criteria. We are confident vast additional improvements can be made.

Especially exciting is the prospect of applying the recent idea of Generative Adversarial Networks (Goodfellow 2014) to speech synthesis. GAN’s learn a complicated loss-function in parallel with the generative model’s parameters, and have been shown to perform well precisely on the open-ended, complex, non-convex tasks that Popa argues include the problem of voice morphing.

We’ve drawn a few non-algorithmic tools from the literature, too: Popa et al test on the CMU Arctic dataset, confirming that our data acquisition efforts are on the right track. And the Ye slides suggest some good quantitative metrics for voice similarity, for instance “Spectral Distortion”.

In short, the prior work provides insight and inspiration not only into techniques applicable to our problem, but also into the challenges that remain to be solved.

## 2. Data

*We chose it because we deal with huge amounts of data.*
*Besides, it sounds really cool.*
--- L. Page

### *2.0. Large official datasets*

We will use the freely available CMU Arctic dataset, as well as the Buckeyes Natural Conversation Corpus.

ARCTIC consists of paired speech clips of ~1000 sentences, each read by the same 7 humans. Of those 7 humans, 3 speak with standard american accents, and 4 speak with ``other accents'', providing variety suitable to training a speech-transcriber.

The Buckeyes Corpus, a larger research database, consists of ~300 tracks of natural conversation. Each track records ~10 minutes of a single speaker in an in informal interview setting.  The 40 speakers recorded in this corpus were residents of central Ohio, were middle-class, and were white speakers, but they varied in gender and age.  The number of speakers was chosen to reduce the effect of interspeaker variability (Pitt et al. 2004).  Tracks have complete phonetic transcriptions, with each phoneme timestamped to the microsecond; at a rate of ~5 phonemes per second, this yields ~3000 (audio segment, phoneme) pairs per track. In total, then, we have ~50 hours of labeled speech, and ~ 900,000 labeled phonemes.

### *2.1. Homemade Recordings*

In addition to downloading standard linguistics datasets, we are also building our own problem-tailored dataset for analysis and preliminary training. Here, we control voice and text (style and content), alternately fixing one to isolate variations in the other. We were able to record and pre-process ~25 records in an hour, a rate which scales to our goal dataset size of low hundreds. Below, we copy nearly verbatim our documentation for a size-95 set of voice samples we've taken of ourselves:

#### *2.1.0. What filenames mean*

Each file is a stereo .wav containing at least one recording of a fixed speaker using a fixed voice style on a fixed text. File durations range from ~4 to ~60 seconds. We recorded in different rooms at different times; each room-time pair has a letter identifier (A, B, ...).

Consider the file name "Sam_nasal_quickfox_3_A.wav". This indicates the speaker (Sam) using a voice (Sam's fake nasal voice) uttered the "quickfox" sentence

("The quick brown fox jumped over the lazy dog.") 3 times in recording session A. See the appendix for source texts used.

### *2.1.1. Dataset size*

We made 23 recordings of "The quick brown fox..." in recording session A, and 72 further recordings (of various texts) in recording session B:

| A&B | Standard | Noisy | Nasal | Deep | Total |
|---|---|---|---|---|---|
| Heming | 2 | 3 | 1 | 1 | 7 |
| Ian | 0 | 0 | 0 | 0 | 0 |
| Sam | 27 | 0 | 27 | 27 | 81 |
| Seth | 2 | 3 | 1 | 1 | 7 |
| Total | 31 | 6 | 29 | 29 | 95 |

### *2.1.2. Recording process*

Participants read texts such as "The quick brown fox jumped over the lazy dog", an accidentally altered version of the classic sentence that contains all 26 letters (and hence an unusually broad range of phonemes). Speakers are instructed to read *clearly* and at *moderate speed and volume*.

The recordings are performed on Sam's Chronos 7 laptop using Audacity. Speech occurs 1 to 2 feet away from the screen, facing the screen, unless otherwise specified. Our 4 voice styles are defined as follows:

0. [standard] --- ordinary clear speaking voices. This is the voice style to which speakers default in daily life, modified by the requirement for "clarity". Thus, good articulation replaces mumbling and slurring.
1. [noisy] --- in these and only these recordings, speech occurs at distance roughly 4-6 feet from screen. This emphasizes background noise. Other than background noise, this style should match [standard].
2. [nasal] --- fake nasal voice. Nasality is achieved (unconsciously) by lowering the velum, hence increasing airflow through the nose, for instance by clenching the jaw or bunching the back of the tongue.
3. [deep] --- fake deep voice. Depth, in contrast to nasality, is achieved by enlarging resonant cavities, for instance by relaxing the throat and

depressing the tongue. This emphasizes lower frequencies.

Post-processing involved:

0. Manual removal of extraneous sounds such as inter-record clickings and voice-work instructions.
1. Amplification (maximum possible without clipping. This is a standard Audacity tool with no parameters).
2. Noise reduction (with the standard Audacity settings, namely (12, 6.00, 3) for (decibels, sensitivity frequency smoothing)).
3. Manual splicing of "silence" to align each record within a single file to be spaced roughly every N seconds, where N is 4, 10, 12, 15, or 20, as appropriate to the text. Since each file has at most 3 records, each file is less than 60 seconds long.

### *2.1.3. Recording Sessions*

Recording Session A: Shapiro 1152 on 2016-11-03 (Thursday) 19:00-20:00.
A small, enclosed room with quiet but audible air conditioning.

Recording Session B: Stockwell 3000 on 2016-11-13 (Sunday) 14:30-13:30.
A larger enclosed room with a typist in the background, a window looking over an occasionally audible street, and an array of hard surfaces that enable echoes.

### *2.1.4. Future work*

We hope to expand the dataset in several directions. Ranked in descending importance:

0. new speakers
1. new sentences
2. more records of current speakers/voice styles/sentence(s)
3. new voice styles

Also important will be a distinction between voice timbre and phrasing; these aspects of voice style are currently treated as one, but the former is local, the latter global, and hence our system will likely treat the two as different.

We also plan to standardize alignment times for each text.

### *2.2. Data Analysis*

We here examine and discuss spectrograms of our own speech. Each team member read aloud “The quick brown fox jumped over the lazy dog” in voices ranging from “nasal” to “standard” to “deep”, with multiple trials per voice (see Section 2.1 for data-acquisition details).

#### *2.2.0 Summary*

Our main observations we summarize as follows:

- As seen in class, visual inspection of spectrograms more easily reveals segmental information (content) than voice (style). Indeed, different speakers’ spectrograms are indistinguishable from afar.
- White noise has a uniform-magnitude spectrogram. Upon noise reduction via standard tools, gaps between harmonics become defined.
- Obstruents such as plosives and fricatives have distinctly taller and more evenly spread frequency components than sonorants. Thus, the “x”, “j” and “z” stand out as peaks in our recordings of the “quick fox” sentence.
- Harmonic dispersion seems associated with various impressionistic qualities like “brightness”, nasality, “richness”, and “depth”. Since perfectly periodic signals should have perfectly banded spectrograms (“harmonics”), we interpret the harmonics separation as a measure of periodicity. Therefore: periodic signals sound “bright”.

The above observations are supported by data in the following discussion.

#### *2.2.1 Details*

We begin with the spectrograms of our standard voices:

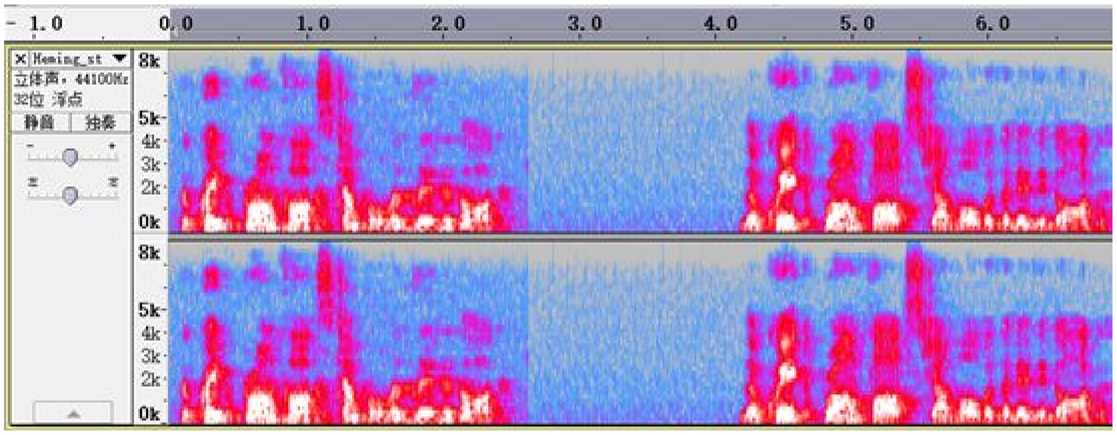


Figure 1: Heming – standard

The spectrogram of Heming Han – standard contains two times of speaking "the quick brown fox jumped over the lazy dog."

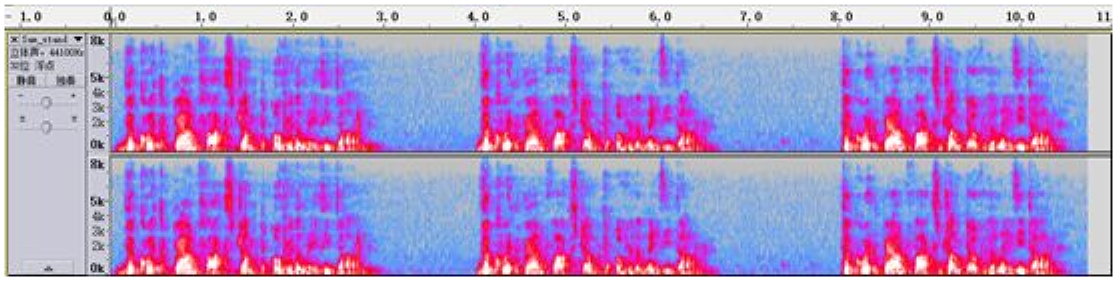


Figure 2: Sam – standard

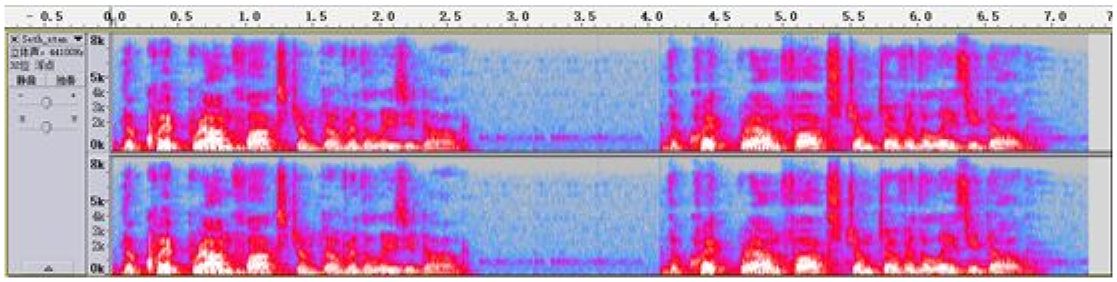


Figure 3: Seth – standard

Above three spectrograms are all standard speech, so we can use them as references when compared with nasal/ deep voice of a single group member.

Our initial observation is that white noise is reflected as a uniform-magnitude addition to the spectrogram. We are able to eliminate the environmental noise from the original sound track, to improve the audio quality for higher precision of possible recognition.

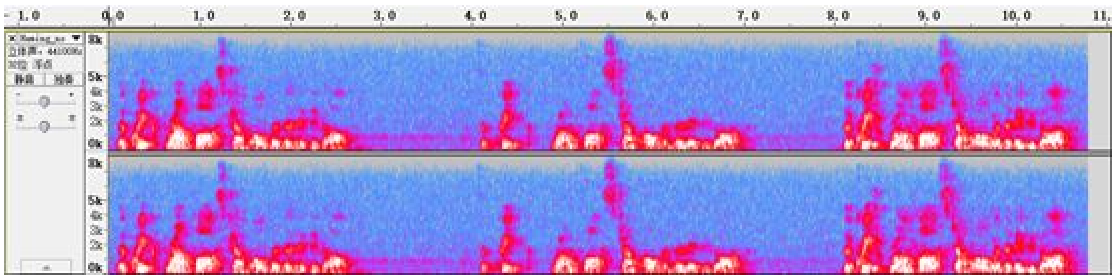


Figure 4: Heming – noisy

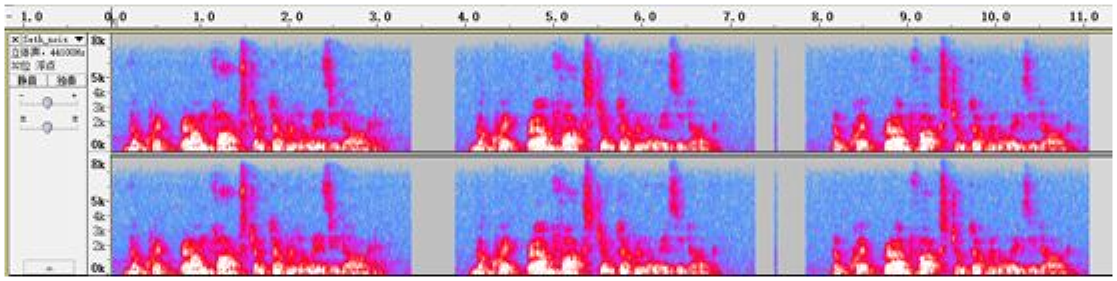


Figure 5: Seth - noisy

What is the influence of noise? We can compare our spectrogram in Figure 1 and Figure 4, as well as Figure 2 and Figure 5. What we can discover is that in high frequency region (frequency that is higher than 3kHz), the resolution of noisy sound file is much poorer than standard non-noisy sound. The reason is obvious: the intensity of high frequency harmonics of human voice is far lower than the intensity in 0~3000Hz. By subtracting the noise of the environment (recorded sound when nobody is speaking), we can get a sound file with higher high frequency performance. Besides, according to what we do from previous assignments (comparing the piano notes with single sinusoidal sound), the higher frequency harmonics preserve the sound quality of our voice. The elimination of the noise, makes it available for us to carry out a transformation between different style.

The second part is the difference between different people's voice. Pay attention to Figure 1 and Figure 2 with Figure 3. There is a very interesting point and we make some assumption: the gap (or the richness) in high frequency domain are different: the record of Seth, whose sound is more deep and rich in daily life, has less gap in frequency; while for Heming and Sam (especially Sam because Heming is not a native speaker and his poor performance in high frequency domain may due to the pronunciation skill), the more light/ bright sound results in a less periodic sound. By comparing the continuity of the spectrogram in higher frequency, we (or computer) may catch the difference of voice style, make it possible to find out the style difference and do something about it.

To make sure there do exist some difference in continuity in high frequency if the sound sounds different in richness, we also record the same sentence trying to use deep voice and nasal voice. One thing worth attention is that except Sam, the record for nasal voice only works for first several words because we are not able to keep using nasal sound. Following are our records:

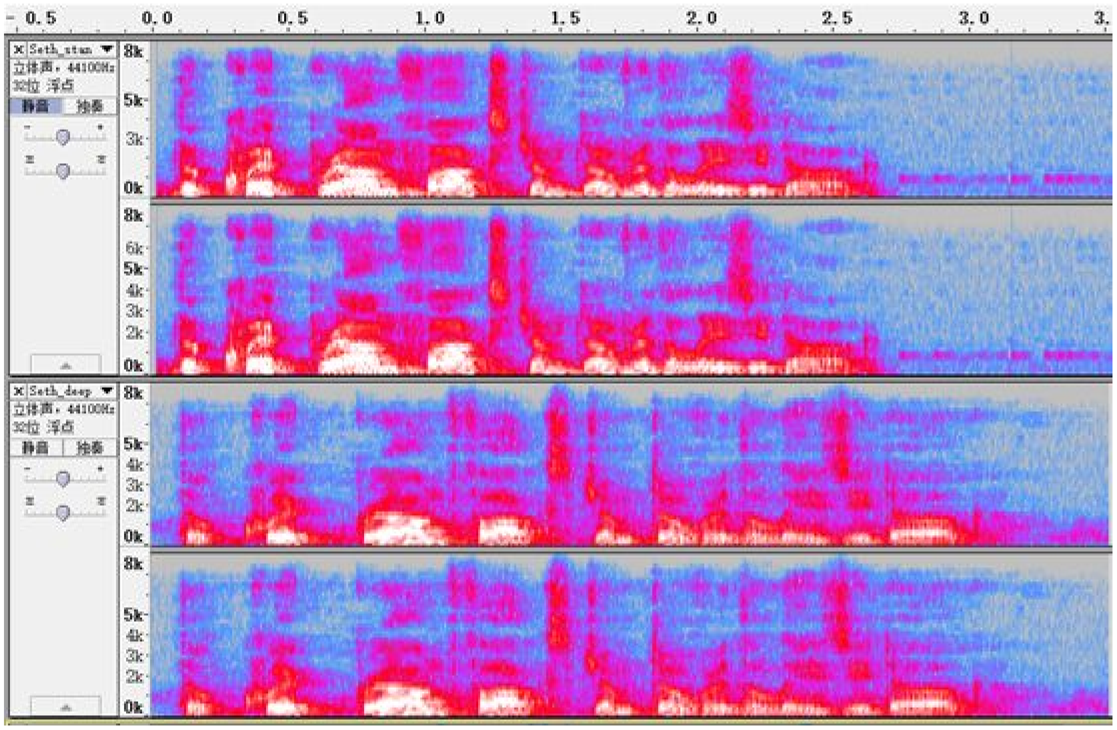


Figure 6: Seth – standard vs. Seth - deep

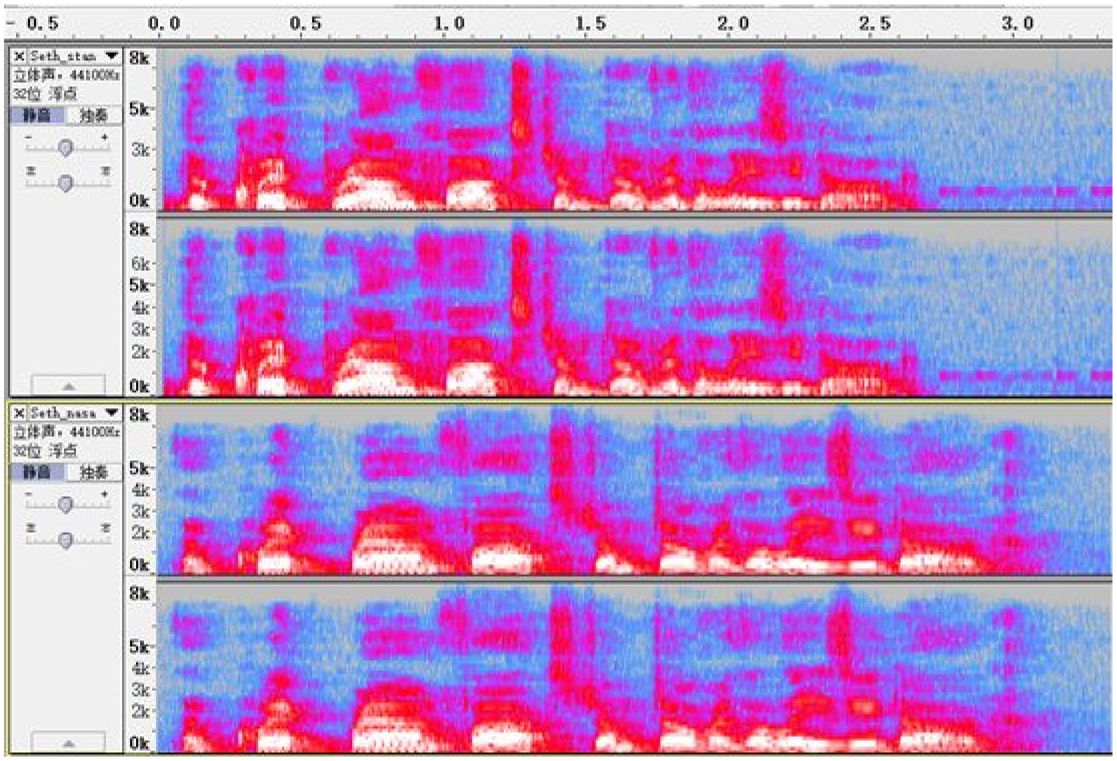


Figure 7: Seth - standard vs. Seth - nasal

What we can see from above two figures are:

1. Compared with standard speech, nasal speech has more gap (discrete frequency pattern).
2. Compared with standard speech, deep voice does not differ much from standard speech, though deeper voice does have a more extensive distribution of frequency. Actually, when we try deep voice, we only tend to lower our frequency as well as making our voice full. As a result, we may conclude that the frequency does not affect much about the overall pattern of the spectrogram; only the quality, the richness matters. By comparing the degree of their frequency distribution of spectrograms, a computer can get the style.

Following are Sam's comparison. It proves our thought.

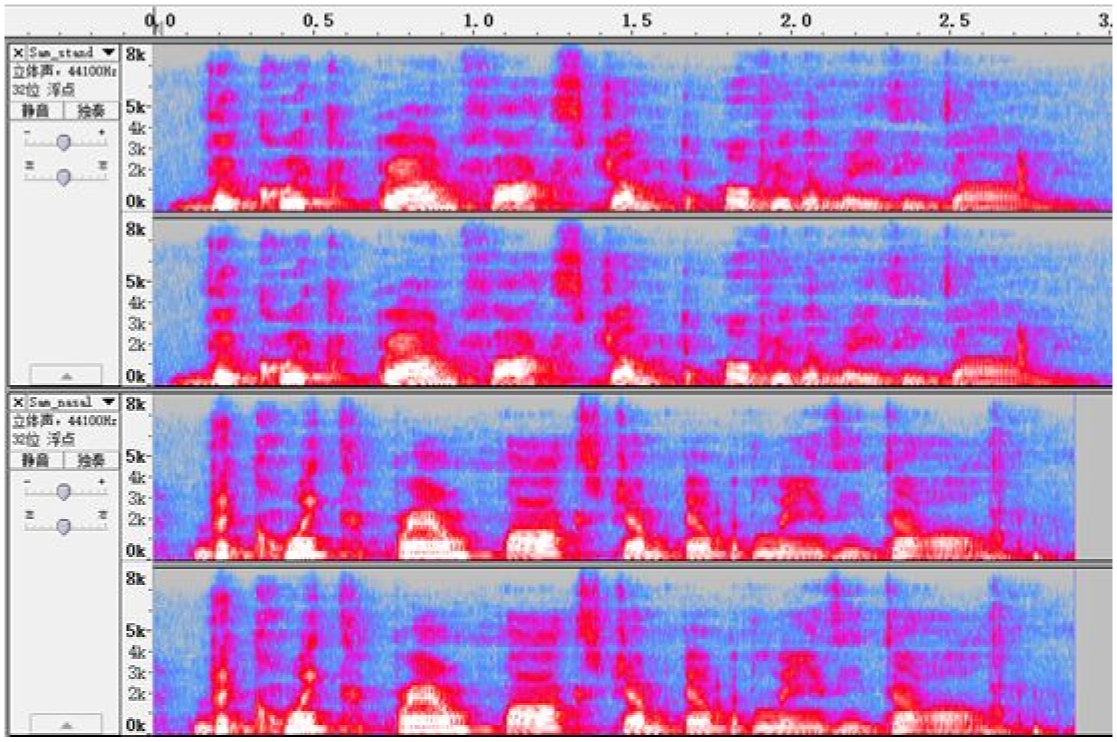


Figure 8: Sam - standard vs. Sam – nasal

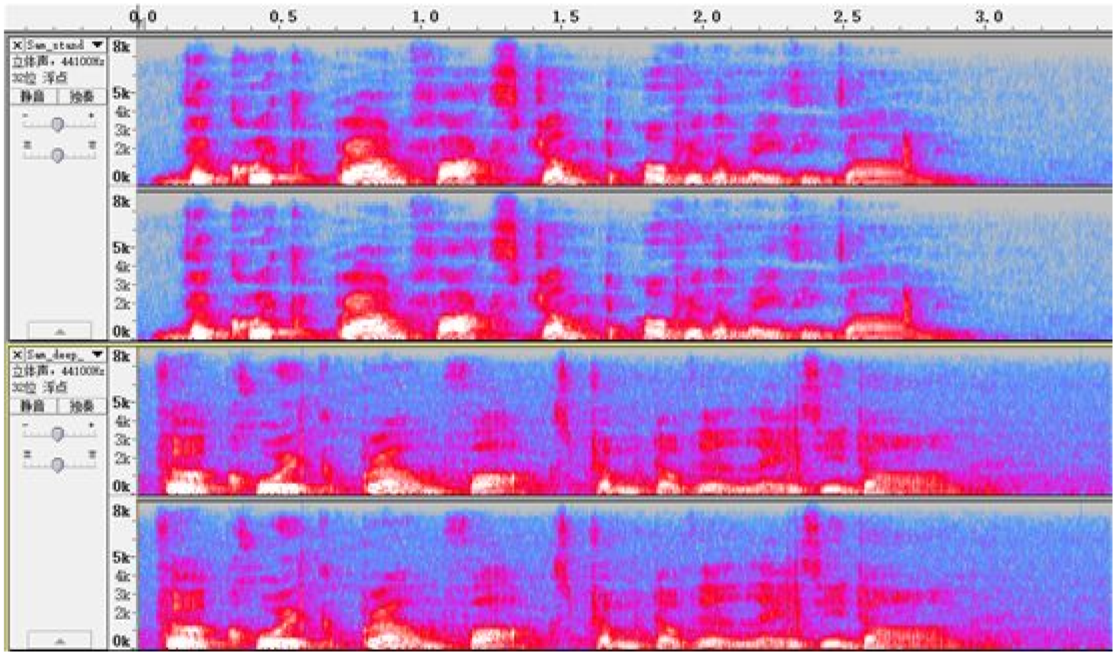


Figure 9: Sam - standard vs. Sam-deep

Third part is the frequency analysis for words. We can see that though we three have totally different voice quality, though we are talking about the same sentence, we have a very similar pattern of the spectrogram.

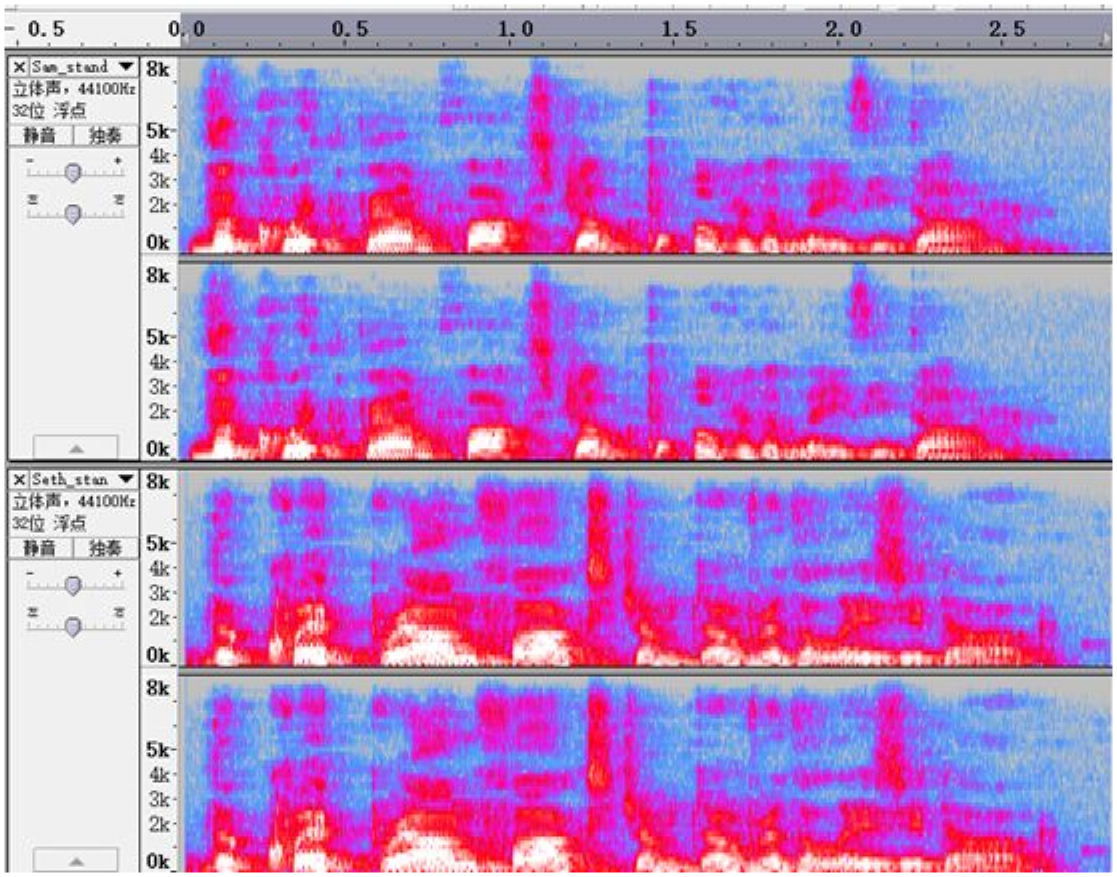


Figure 10: Sam vs. Seth

What we can conclude from our figure and our record is that: the most intensive low frequency domain (which is shown as white in our figure), is determined by the vowel; “th” in “the”, “n” in “brown”, “j” in “jump”, “x” in “fox”, “z” in “lazy”, in our spectrogram, has a very extensive frequency distribution and easy to tell. We believe the computer would have the ability to tell them and achieve recognition.

In conclusion:

For words, we can look at the acoustic characteristic, then compare with what we can get in the database to identify the letters/letter-groups (, then may go to the meaningful word).

For style, one interesting discover is the gaps/continuity/distribution in harmonics: brightness versus richness.

## 3. Validation

*Nearly all men can stand adversity, but*
*if you want to test a man's character,*
*give him power.*
--- A. Lincoln

We seek to validate our system not only end-to-end, but also component-wise. In designing a validation system, our main challenge is the absence of an obvious metric on outputs: there's no easy formula for how close one audio sample is to the other, psychologically; and when we try to compare intermediate representations such as segmentations and summaries of voice qualities, the notion of "good vs bad" becomes yet more ill-defined. We here summarize the metrics we plan to use.

As a matter of hygiene, we will split our dataset into a *visualization*, *train*, and *test* set. To avoid rampant overfitting, the test set will be used for and only for validation of our system; in particular, we will train neither our algorithms nor our brains on this set. The train set will be used and only used by our algorithms. The visualization set will be used and only used by our brains.

### *3.0. Validating Segmenters*

*How do we test a segmentation algorithm?* We seek a function that takes two segmentations and outputs a non-negative real number representing the segmentations' distance. We'd like for small offsets to be penalized less than large ones, and we'd like to be merciful of many-to-one or one-to-many errors. An initial perusal of audio segmentation literature yielded no standard evaluation metrics that convincingly satisfied those constraints.

We thus propose a new metric as follows. Let P be the set of finite subsets of the open interval $(0, 1)$. We view each element x of P as a segmentation of $(0, 1)$: the elements of x are precisely the boundaries of the corresponding segmentation. Define the energy $U(x)$ of each x in P as the sum of the squared lengths of segments, and set

$$d(x, y) = U(x) + U(y) - 2\, U(x \cup y)$$

It turns out that d is a metric on P: for instance, triangle inequality is equivalent to:

$$U(x) + U(y) \leq U(x \cup y) + U(x \cap y)$$

a pretty statement provable by induction on the number of elements in $x \cup y$.

The distance function d, defined as above, ranges in `[0, 1)`. If we extend the formula to compute distances between finite subsets of a length-L interval, then the distance will range in $[0, L^2)$. Moreover, if x consists of S evenly spaced segments, then d(x, y) ranges in $[0, (L/S)^2)$. So we normalize our distance function as follows: if x is a ground-truth segmentation with S segments and y is a generated segmentation, we say

$$\text{dist}(y \text{ from } x) = (S/L)^2\, d(x, y)$$

Note that the dist is asymmetric: `dist(y from x)` need not equal `dist(x from y)`. To evaluate a segmentation algorithm, we compute

```
score = average dist(generated from ground truth)
```

over a test set of (audio, segmentation) pairs. The lower, the better, and a score of 0 is achieved by and only by an algorithm that produces perfect segmentations.

### *3.1. Validating Classifiers*

We will evaluate our phonetic classifiers based on three metrics. The initial two are **selectivity** and **sensitivity**. Plainly, selectivity measures the model's capacity to tell "the whole truth" while sensitivity measures whether the model's capacity to tell "nothing but the truth". Precisely: sensitivity is the ratio of correctly classified tokens to the total number of guesses, and reveals which segments the classifier finds most difficult to identify. Selectivity, 1 minus the ratio of the incorrect guesses to total number of guesses, reveals whether the classifier is actually classifying a segment correctly, or merely frequently guess this segment.

The final measure, **categorical cross-entropy**, is useful for classification algorithms that return probability distributions over the possible classes. Because it has access to richer information --- namely, the "distribution" or "thoughts" of the algorithm --- CCE is able to provide richer feedback, and in some instances accelerate training. For instance, it more greatly penalizes algorithms that confidently predict an incorrect class than it does those that make mistaken predictions less confidently.

CCE thus makes distinctions invisible to sensitivity and selectivity measures. It turns out that the CCE is minimized precisely by the algorithm that returns all perfect answers, a property not enjoyed by either sensitivity or selectivity in isolation. On the other hand, CCE does not distinguish between false-positives and false-negatives; for this, we will rely on selectivity and sensitivity.

### *3.2. Validating Embedders*

See Section 1.1 for a discussion of how GANs partially solve this validation problem.

## 4. Implementation

*The whole difference between construction and creation is exactly this:*
*that a thing constructed can only be loved after it is constructed;*
*but a thing created is loved before it exists.*
--- C. Dickens

*Actions speak louder than words.*
--- Unknown

### *4.0. Speech Segmenter*

We are trying several approaches in audio segmentation. Currently, the most promising method identifies segment boundaries by locating dips in amplitude. In detail, if $S_d$ denotes a linear gaussian low-pass filter of width d, and if $P^p$ denotes a pointwise power $|x|^p \leftarrow x$, then our method selects strict local minima of z, where:

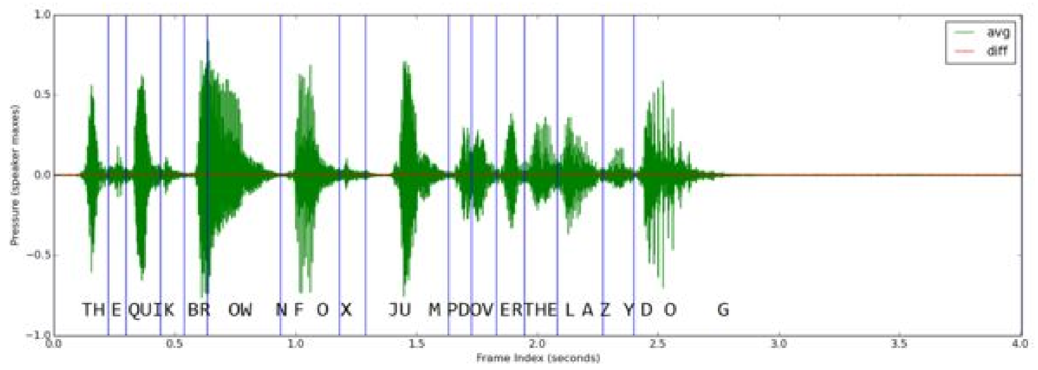
$$y = S_a(P_2(\text{signal}))$$
$$z = S_b(P_{1/2}(\max(y, s\ S_c(y))))$$

If $s=0$, then this reduces to the nice expression $z=S_b(P_{1/2}(S_a(P_2(\text{signal}))))$. The P's introduce nonlinearity. We interpret the four parameters as follows:

| | |
|---|---|
| a (10 milliseconds) | low-pass timescale for raising valleys. |
| b (10 milliseconds) | low-pass timescale for lowering peaks. |
| c (1000 milliseconds) | timescale over which to measure relative amplitudes for silence-detection. |
| s (0.1, unitless) | relative amplitude threshold below which signal is deemed to be silent. |

The silence parameters c and s control allow us to avoid spurious minima at what are actually near-silent times by smoothing those silent areas, removing all minima.

An example segmentation using the above approach is as follows:



The green signal represents a normalized audio amplitude --- the system's input.The red line shows the stereo difference between left and right; here, there is no visible difference. The blue vertical lines denote the boundaries produced by our segmentation algorithm. We hand-annotated the above with a phonetic transcription of the spoken sentence. Observe that the system identifies every syllable boundary, and that every boundary it identifies is a syllable or phoneme boundary. While some phoneme boundaries (BR/OWN; FO/X; LA/ZY) are identified, others (B/ROWN; D/O/G) are not. We conclude that the system had perfect precision and moderate recall on this data point.

### *4.1. Speech Classifier*

The speech classifier will be implemented as a Long Short-Term Memory RNN. The creation of the network is divided into two portions - preparing the transcribed audio to be used as input to the neural network, and the training of the model.

Luckily, the Buckeye Corpus is not only fully phonetically transcribed, but the wav file timestamps corresponding to the start of certain segments is given to us. With this info, I import the wav files, segment the files, window short overlapping slices (currently, windows of length 60ms and with time step of 30ms) of each segment (using a rectangular window at the moment), and compute a DFT of each of these windowed clips. Because nets output numerical values, not characters, directly, I converted the transcribed segments to unique numbers. This meant that the for each sound file, there is now a list of 2-tuples, where the first element is the DFT of windowed clip, and the second element is a unique number corresponding to the output segment.

I am working on training the model now using the LSTM architecture provided under the Python library Keras.

### *4.2. Speech Embedder*

We attack the speech embedding problem from multiple angles. On one hand, we implement expert-crafted features such as the MFCC coefficients discussed below. On the other hand, we will explore deep learning methods to learn deeper embeddings.

The speech embedder is basically based on Mel-frequency cepstral coefficients. It is a common method used nowadays to achieve speech recognition. The main idea of this method is to build a set of filters and a set of frames(audio segments), then apply these filters on these frames, get a corresponding set of coefficients evaluate the target audio performance in in each filter, to achieve the goal of recognition.

MFCC contains several main steps:

0. Frame the signal into short frames -> matrix with signals in matrix.
1. For each frame calculate the periodogram estimate of the power spectrum -> basically averaging over each segment to get the overall characteristics for each frame (lossy).
2. Apply the mel filterbank to the power spectra, sum the energy in each filter -> draw characteristics from the results of applying these filters, to preserve only the information we need for recognition (lossy).
3. Take the logarithm of all filterbank energies -> get cepstral coefficients.
4. Take the DCT of the log filterbank energies, discard some of the coefficients.

Fortunately, there is some python implementation of MFCC method online, using packages like numpy and scipy. We are now working on understanding the code provided online, try to invert the method to accomplish synthesis (coefficients to audio), and hopefully make our own version of MFCC recognition implementation. Since the MFCC method is lossy so that the invert is a worldwide hard problem nowadays, we do not expect a high quality --- we expect to generate some recognizable voice segments.

Currently we are working on understanding the building of filter bank and the invert of the last step.

## 5. References

*In the end, we self-perceiving, self-inventing, locked-in mirages*
*are little miracles of self-reference.*
--- D. Hofstadter

We learned about formants:

- Formant (Wikipedia Article, 2016-11-03)

We read about the Voice Morphing work of the past 1.5 decades from:

- High Quality Voice Morphing Seminar (Ye 2004 --- Slides)
- High Quality Voice Morphing (Ye and Young 2004)
- Local Linear Transform for Voice Conversion (Popa et al. 2012)

We'll train and test on:

- CMU Arctic: Databases for Speech Synthesis (Komenek & Black 2003)
- Buckeye Speech Corpus

We're excited about these algorithms to treat style/content separation:

- The cepstrum: A guide to processing (Childers et al. 1997)
- Generative Adversarial Networks (Goodfellow et al. 2014)
- Long Short-Term Memory Recurrent Neural Networks (Hochreiter & Schmidhuber 1997, Graves 2013)
- A Neural Algorithm of Artistic Style (Gatys et al. 2015)
- Learning What and Where to Draw (Reed et al. 2016)

Segmentation Evaluation Metrics:

- Rich Transcription Meeting Recognition Evaluation Plan (NIST 2009)
- Audio Segmentation for Speech Recognition using Segment Feature (Rybach et al. 2009)

## 6. Appendix A: Source Texts

The following 8 texts, selected as sources for our home-made speech dataset. cover a wide range of sounds and writing styles.

### *6.0. Classic Pangram*
The quick brown fox jumped over the lazy dog.

### *6.1. From "Jabberwocky" by L. Carrol*
`Twas brillig, and the slithy toves
    Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

### *6.2. From "The Raven" by E.A. Poe*
    And the silken, sad, uncertain rustling of each purple curtain
Thrilled me—filled me with fantastic terrors never felt before;

### *6.3. From "Good Old-Fashioned Lover Boy" by Queen*
I can dim the lights and sing you songs full of sad things
    We can do the tango just for two
I can serenade and gently play on your heart strings
    Be your Valentino just for you

### *6.4. From "Hamlet" by W. Shakespeare*
O, that this too too sullied flesh would melt
    Thaw and resolve itself into a dew!
Or that the Everlasting had not fix'd
    His canon 'gainst self-slaughter! O God! God!

### *6.5. From "Lovers of the Poor" by G. Brooks*
The stench; the urine, cabbage, and dead beans,
    Dead porridges of assorted dusty grains,
The old smoke, heavy diapers, and . . .
    The darkness. Drawn
Darkness, or dirty light. The soil that stirs.

### *6.6. From "A Mathematician's Apology" by G.H. Hardy*
The mathematician's patterns, like the painter's or the poet's must be beautiful;

the ideas like the colours or the words, must fit together in a harmonious way.

***6.7. From "Adventures of Kavalier and Clay" by M. Chabon***

Sammy performed the rapid series of operations-which combined elements of the folding of wet laundry, the shoveling of damp ashes, and the swallowing of a secret map on the point of capture by enemy troops-that passed, in his mother's kitchen, for eating.