MLND Capstone Project

Sub-Vocal Recognition, an Articulatory Feature Extraction Approach

Brian Coe

June 28, 2017

I. Definition

Science fiction has described numerous characters who communicate through thought alone. From H.G. Wells' "Time Machine" to Isaac Asimov's Foundation novels to William Gibson's "Neuromancer", artificial telepathy seems an integral part of any high-technology future. Hopefully, recent developments in machine learning and wearable computing hardware will breathe new life into the seed of such an exciting future.

Sub-vocal recognition (SVR) uses electromyographic (EMG) recordings from the muscles of speech to recognize spoken or sub-vocalized language. Electrodes are usually placed on the skin of the throat during speech or reading. During otherwise "silent" reading, EMG data can still contain enough information to decode what the user was reading to themselves. A common approach is to use EMG data from vocalization to train recognition systems for sub-vocal (silent) recognition. An early NASA project used a variety of approaches, but found high variance between speakers and for a single speaker on different days. High variability lead to accuracy ranging from 90%+ one day, down to 48% the next, for a single speaker. [1] A later study by Carnegie Mellon University focused on developing feature extractors based on unique linguistic aspects of phonemes, called articulatory features (AF's). [2] This later approach achieved F1 scores as high as 0.45 for some phonemes, with EMG data from multiple facial locations.

Connecting Neural Activity to Unique Phoneme Features

As mentioned, even silent reading can produce enough motor activity to register on an EMG. Such activity, known as sub-vocalization, is a promising entry point for artificial telepathy. Sub-vocalization often happens subconsciously during reading, but can also be deliberate. A region of the throat known as the pharynx contains muscles and nerves involved in all manner of human speech. With sufficient sensitivity, accurate sub-vocal recognition through careful electrical measurement of this single region should be possible. Linguists have categorized phonemes on the basis of AF's, with four dimensions in particular as the most common: place, manner, height, and vowel. Each of these AF's correspond roughly to specific actions of the muscles of speech. "Place", for instance, correlates to the area of the mouth held in tension to produce a phoneme.

Specific muscular actions, like holding a body part in tension, requires neurons to "fire" in specific patterns. When neurons fire, the pattern of voltage discharge they produce is known as an action potential. When action potentials from motor neurons activate muscle, they are known as motor unit action potentials (MUAP's). [3] By carefully deciphering MUAP's from EMG data, it may be possible to detect distinct signatures for the different kinds of AF's, and the values they can take on (in the case of "place": dental, labial, alveolar, etc.). Detecting the common features of disparate phonemes in this way may help improve the accuracy of an SVR phoneme model when compared to using a less domain-specific approach that doesn't benefit from phoneme commonalities.

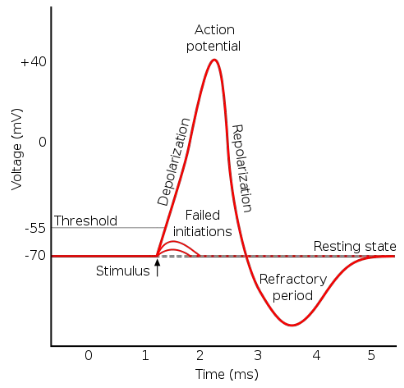


Illustration 1: Conceptual graph of individual neuron action potential

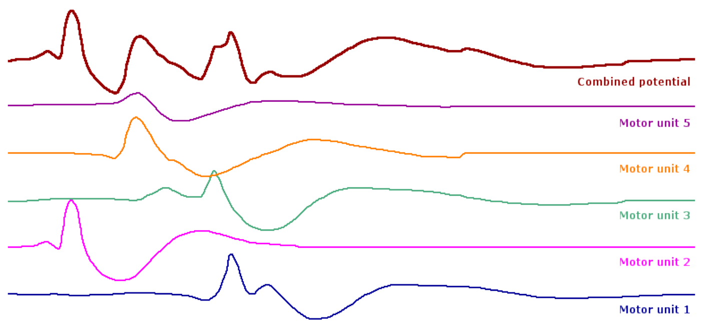


Illustration 2: Superpositioning of individual MUAP's produces the complicated values of EMG data

Distinct MUAP's arise from different actions by different muscles in different locations. Therefore, the MUAP's spatial and temporal location, wave width, and intensity of the MUAP can all be used to help distinguish different motor activities in EMG data. Since distinct AF's require different patterns of muscle activity, they should require distinct MUAP's. Correlating specific MUAP's from EMG data to AF's should therefore help identify common and distinct features of phonemes, ultimately aiding in identifying the phonemes themselves.

Previous work from Carnegie-Mellon also used AF's, but their approach was based on multiple EMG sources, AF extractors as Gaussian Mixture Models, the Short Time Fourier Transform, and forced phoneme-EMG alignment by using a signal channel and automatic auditory speech recognition. It should be noted their usage of AF's is based on a binary encoding scheme, with one classifier per unique label, and uses slightly different labels than the present study. This is intended to illustrate the relative simplicity of the present study. The data used herein was generated by the author using a raspberry pi computer with an 8-bit analog-to-digital converter (ADC), two monopolar surface electrodes as a single differential voltage measurement source, and no forced time alignment or audio signals.

Data from EMG during full vocalization is often used in addition to sub-vocalization in training EMG-to-phoneme models. Full and sub-vocalization EMG signals vary mostly by absolute intensity, not morphology, given they arise from fine motor units. Higher vocal volume mostly corresponds to increased speech muscle contraction intensity, leading to MUAP's with higher amplitude, but a similar overall width and form to MUAP's from quieter vocalizations. Ideally, EMG signals from full and sub-vocalization are isomorphic to one another via a simple additive or multiplicative transform. Data from full vocalization should therefore have a higher signal-to-noise ratio on low resolution hardware in noisier environments, easing feature extraction and model training. In particular, between full and sub-vocalization, energy spectra should vary in absolute terms more than in relative terms among phonemes. Energy spectra differences between phonemes gleaned from full vocalization EMG data should therefore be useful in identifying sub-vocal EMG data as well.

Transient signals from MUAP's in EMG data vary in width and intensity. For a given kind of fine motor contraction with a fairly constant duration, we would expect the EMG signal to have a fairly constant width, but vary in intensity or amplitude between samples at different contraction strengths. Wave amplitude should therefore correlate to contraction strength which in turn correlates to vocalization volume. Full vocalization data should have a higher overall spectral energy, but the relative spectral energies of phonemes should be similar to sub-vocalization. Increasing the total recorded spectral energy should reduce the effects of noise and accentuate relative phoneme differences. The energy input from most noise sources should be independent of the energy of the recording target, making full vocalization a method of increasing signal gain.

Full Vocalization as Foundation for Sub-vocalization

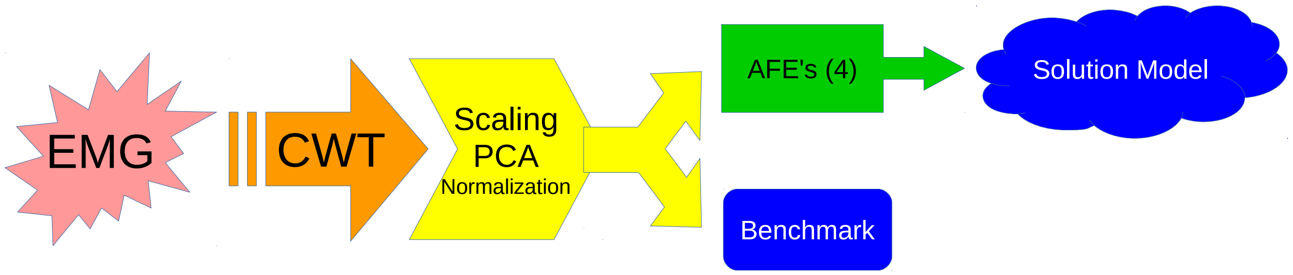
Processes and methods developed while using full vocalization to identify phonemes in EMG data should transfer well to sub-vocalization scenarios. The two recording conditions differ mostly in the sensitivities required to distinguish phonemes, but general relative patterns among phonemes should hold. The models trained in this manner, using full vocalization, either can themselves be used when higher quality data becomes available, or the general process used to obtain these models can be improved upon and re-applied using improved recording equipment. The former case, use 'as-is' seems unlikely given current results, as the models might not be able to take full advantage of the new equipment, while the latter case would constitute boot-strapping for future process improvements. In general, the lower spectral energies in sub-vocalization EMG data could still be distinguished in a live setting by a sufficiently well-trained model (one trained on full vocalization EMG data, for instance).

Defining the Problem

Ahead, then, lies the challenge of developing an EMG-to-phoneme pipeline, with specialized feature extraction models to aid in identifying phonemes from processed EMG data. EMG data in its raw form is not particularly useful, but techniques like wavelet transformation can help accurately extract information about the energy spectrum of the non-stationary, non-sinusoid waves it records. Next, information about AF's is extracted from these energy spectra by models specialized for the purpose. Ultimately, a phoneme prediction model will use the extracted AF's and supplemental energy spectra data to identify the actual phonemes used to generate the data.

Solving the problem will proceed by recording data, transforming it, training a benchmark model, training AF extractor models, and finally: training and optimizing a phoneme model. Recording the EMG data will be done using an ADC with a raspberry pi. Recorded EMG data will be visualized and statistically analyzed in attempting to determine the separability and salience of phonemes in the data. Transformed EMG data will undergo a similar analysis, to determine if employed transforms are appropriate for enhancing phoneme salience and separability. A benchmark model that predicts phonemes based on EMG without using any kind of AF's will be trained and optimized to set a performance goal for the AF extractors and phoneme model using them. Next comes training and optimization of the AF extractor (AFE) models themselves, using a grid search. Lastly, the EMG-phoneme model using AF's, the intended solution, will be trained and optimized by grid search.

Ultimately, the intended solution consists of AFE models specialized to find AF's from processed EMG data, which feed their predictions into a phoneme prediction model. The phoneme prediction model uses predicted AF's alongside processed EMG data to detect phonemes in a given data segment. Ideally, this AF-enhanced phoneme model will outperform the benchmark model lacking AF extraction.



Drawing 1: Sketch of intended solution flow. Continuous Wavelet Transform is done on EMG data, scaled, reduced, and normed. Benchmark and AFE models are trained. AFE outputs serve as inputs for the Solution Model

Metrics

Performance in this case is measured by F1 score. Sklearn, the framework used here, gives most estimators a built-in F1 scoring method accessible by '.score()', and this is the method used to compare results. F1 is suitable here because it equally weights precision and recall. Essentially, recall means getting true positives and avoiding false negatives while precision is avoiding false positives. In single phoneme detection, low precision and recall are about equally problematic. Since this isn't a clinical or critical use case, all “misses” are about equal. If or when this solution is rolled into a more general EMG-based speech recognition system, it will be the relative probabilities of phonemes that really matter. Recognition systems have some ability to compensate for low single recall or precision. Relative probability of phonemes should be relatively unaffected by low recall or precision, since the correct phoneme could still be in the top few most likely predictions for a single data window. A full speech recognition solution built with this system should be able to compensate for slightly mis-ordered phoneme probabilities by injecting contextual, a-priori information to improve single phoneme results.

II. Analysis

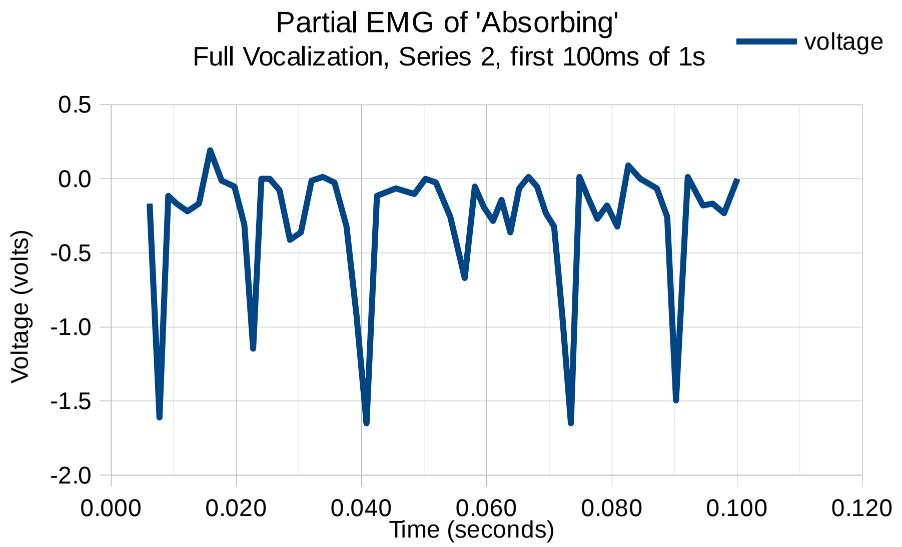


Table 1: Graph of raw voltage data from first 100ms of data in 'absorbing' series 2

time	voltage
0.006	-0.168
0.008	-1.611
0.009	-0.116
0.010	-0.168
0.012	-0.219
0.014	-0.168
0.016	0.193
0.018	-0.013
0.020	-0.052
0.021	-0.309
0.023	-1.147
0.024	0.000

Table 2: Sample of raw voltage values from first few points in 'absorbing' series 2

Data for SVR has been recorded by the author using a Raspberry Pi breakout board based on the PCF-8591 ADC. [5] Recordings were made in differential voltage mode on the 8-bit ADC using 2 monopolar electrodes attached approximately 4cm apart on the right pharyngeal region.

Each word was recorded as a single file. Before recording, the word was entered as a file name with a series suffix. Upon pressing 'enter', recording began, full or sub-vocalization commenced, and when complete, the program and file recording terminated with ctrl-z. This process was repeated for each of about 148 words, three times in total. The first series was recorded during sub-vocalization only, while series 2 and 3 were recorded under full vocalization. A mixture of full and sub-vocalization were used in an attempt to improve model robustness. Each word was recorded as a single column-separated value (CSV) file.

Words used for the EMG recordings were generated by random selection from a dictionary. Each word was decomposed into phonemes using natural language tool kit's 'cmudict', which has about 150k words as keys and a list of phonemes for each word as its values. Phonemes listed there follow IPA standards. Rasipuram was consulted for constructing AF vectors for each basic phoneme. [4] Below is a sampling of how words map to their phonemes, and how phonemes map to AF's, but complete lists for each, respectively, are available in appendix A and B.

advice	['AE' 'D' 'V' 'AY' 'S']
aftermath	['AE' 'F' 'T' 'ER' 'M' 'AE' 'TH']
amazing	['AH' 'M' 'EY' 'Z' 'IH' 'NG']
amuck	['AH' 'M' 'AH' 'K']
animated	['AE' 'N' 'AH' 'M' 'EY' 'T' 'AH' 'D']
aspiring	['AH' 'S' 'P' 'AY' 'R' 'IH' 'NG']

Table 3: Sample of word to phoneme transformations

'AE'	['vowel'	'mid-front'	'low'	'yes']
'D'	['voiced-stop'	'alveolar'	'max'	'no']
'EY'	['vowel'	'front'	'mid-high'	'yes']
'F'	['fricative'	'labial'	'max'	'no']
'M'	['nasal'	'labial'	'max'	'no']
'N'	['nasal'	'alveolar'	'max'	'no']

Table 4: Sample of phoneme to AF transformations

In total, the dataset used for the present results totals 444 files and runs for 350 seconds
in duration. Size of the dataset, not including words, totals 9.0MB in csv format and 5.45 MB when loaded as DataFrames. Average file size and duration is therefore 20.3 KB and 0.788 seconds, respectively. Standard deviation for file memory use is 6.44 KB while the standard deviation for file duration is 0.244 seconds. Words used to generate the data, numbering 148, require about 1.4 KB of memory to store as a python list. On average, the words used are 6.07 letters long, 5.03 phonemes long, with standard deviations at 1.95 and 1.95, respectively. The average voltage recorded in the EMG data across all files is -0.339, with a standard deviation of 0.0493. For a closer look at EMG recording variability (specifically: in the second series), see the table.

Word	Average V	STD V
equable	-0.301	0.418
amazing	-0.304	0.425
equable	-0.301	0.418
motion	-0.277	0.392
cakes	-0.324	0.414
squirrel	-0.302	0.409

Table 5: Sample of average voltages and standard deviations of random words from series 1 data

Particularities

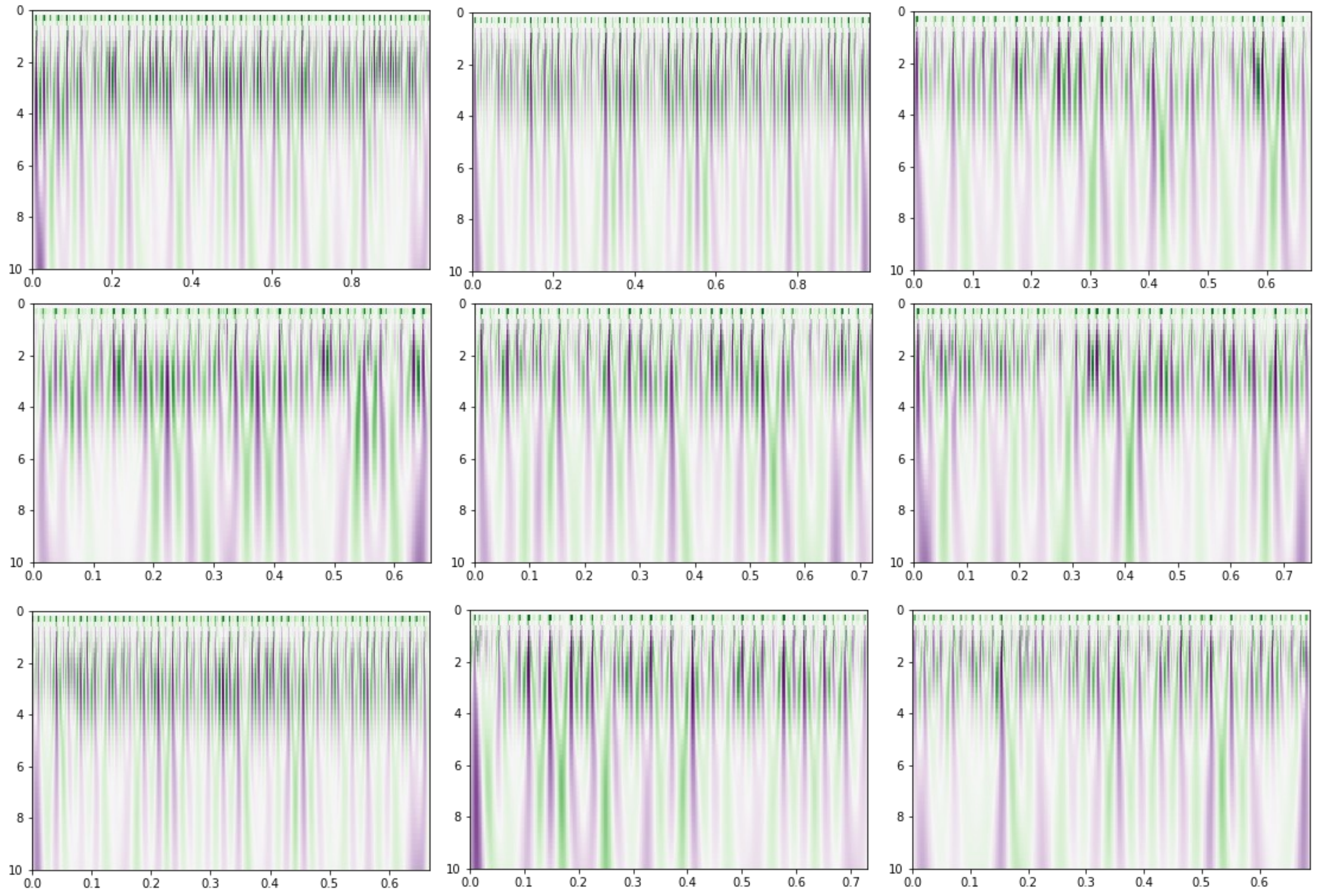
Raw EMG data is not in a particularly useful form for most purposes. To be useful, information about the MUAP's it records needs to be extracted and analyzed by some method. Often, the method used is the Fast Fourier Transform (FFT) or Short Fourier Transform (SFT). Another common method is the Wavelet Transform (WT) which uses the SFT to extract non-stationary waves from the data, but is more effective than SFT when a wavelet shape is chosen to closely match the waves of interest. Further considerations include whether to use surface electrodes or invasive needle electrodes. Surface electrodes often encounter interference from intervening tissue, in addition to interference along the sensor wires and from the recording environment. Variations in electrode placement and biological variability also effect data collection.

For the present study, the ADC used yields relatively low voltage resolution, offering 8 bits. 8 bits means only 2^8 or 256 different levels of voltage. Compounding this difficulty is the lack of a programmable gain amplifier (PGA) within the ADC, so only a portion of the 3.3 volt range can be used at any time, effectively limiting the number of resolved voltage levels to about 128. Taken together, this is about 512 times lower resolution than in other similar studies, and will be the first aspect to improve in future iterations upon this study. The sampling rate of 600Hz used here is quite close to standard, but signals well over 1kHz will be difficult to resolve. Fortunately, most MUAP's occur in the 10-50Hz range.

In transforming words into phonemes, the cmudict offers variations of phonemes that lack specification of AF's in the other academic sources employed. When these less common varieties of phonemes are encountered, they are recast as the basic type ('A0', 'A1', 'A2', etc. all become simply 'A'). This ensures all phonemes will be treated as having the same AF's, and all phonemes used herein will have complete AF vectors built for them, even if this means introducing some systematic error. Distinguishing among more phoneme varieties and their unique AF's will present yet another direction for future improvement. Some words generated initially were also missing altogether from cmudict, so while there are files for 'direful' and 'wrathfu' in series 1, they were not used in the study and were not recorded for subsequent series. An oversight caused the file 'raspy-1' to be not recorded or missing, while this file is present in series 2 and 3 and those were used in the study.

Exploratory Visualization

Wavelet graphs are visualized in three dimensions: two spatial and one of color. Differing colors show positive and negative phases of wavelets, while the intensity of color denotes amplitude. The vertical axis shows wavelets of differing widths and the horizontal shows time. For a particular wavelet width (i.e., at any particular location along a horizontal line through the graph) the spectral energy carried by wavelets of that width can be found by summing the squares of the absolute value of the calculated wavelet amplitudes.



Panel 1: Wavelet Transform graphs for three samples of three different words. Top to Bottom: Advice, Aspiring, Weather; Left to Right: Series 1, 2, 3. Each graph has time on the X, wavelet width on the Y, color denotes wave phase (purple +; green -), color intensity denotes amplitude.

From the panels above, wavelet timing, width, and intensity can be seen to vary more between words than between samples for a particular word. Additionally, common motifs can be seen for phonemes present within different words. While phoneme-specific patterns can be seen, they can also be seen to have some significant variations in duration and intensity. These apparent wavelet patterns correlate closely to at least a subset, if not all, of the MUAP's responsible for generating the information of interest. As in the wavelets, MUAP's vary in width, location, and timing. Energy spectra obtained from these wavelet transforms should correlate closely to MUAP energies at different wavelengths and distinct times. By learning patterns in wavelet energy spectra, it may be possible to learn to distinguish MUAP's correlated to distinct actions. Actions of interest are those specific to and required to produce the various AF's and ultimately, phonemes. Wavelet energy spectra may therefore help distinguish phonemes by finding AF's that unite some while helping to distinguish others.

Algorithms and Techniques

Algorithms employed to distinguish phonemes in EMG data include the continuous wavelet transform (CWT), word phonetic lookup, AF vector construction, phoneme-based EMG windowing and spectral energy summation, feature scaling, norming and dimensionality reduction (principal component analysis, PCA), and finally, multi-layer perception classifiers with grid searching for optimal parameters. CWT used here has 50 discrete widths evenly spaced from 0.01 to 10 (corresponding to wavelet durations of approximately 20us and 20ms, respectively) and using the Ricker wavelet ("mexican hat wavelet") within numpy's "cwt()" implementation. Words are split into phonemes with cmudict, dataframes of labels built with AF vectors using values from Rasipuram, and the number of windows for each EMG file is the number of phonemes in the word. Data from each file is segmented into a number of equal length windows. Amplitudes for each point within a window are grouped with others of the same width to find their total spectral energy for that window. In other words, after CWT, the 50-valued vectors for each datapoint within each window are condensed into a single 50-valued vector per window. Window lengths vary depending on the file length and number of phonemes in the word, but each window has a single 50-valued total spectral energy vector. PCA is performed to reduce these 50 components per window to 10. Scaling to unit variance is done column-wise and normalization to unit vector length row-wise to help smooth the data and prevent neural net irregularities.

Models employed include a unique MLPC for each AF vector column: manner, place, height, and vowel. Each AFE so trained takes processed EMG as input and yields an AF column as output. A benchmark MLPC uses processed EMG only as input and yields phonemes as outputs. A final model, the solution MLPC, takes processed EMG and AF's generated by the AF extractor models as inputs to predict phoneme outputs. Grid search is performed on each of these six (6) models, with exploratory parameters for layer sizes and alpha. Layer size parameters are explored from two (2) to nine (9) layers with one of either individual layer sizes: sixty (60) or ninety (90) for all layers. Alpha values are explored in the range [1e-4, 1000].

Each algorithm employed is chosen for unique value it brings to the whole system. CWT extracts non-stationary waves from the data. By using a specific wavelet shape, CWT localizes transient waves in time by frequency and intensity. Using a wavelet well-matched to signal waves of interest, the original MUAP's producing the EMG, CWT can outperform FFT and SFT alone. CWT is also more resistant to Gaussian noise than FFT. Splitting words into phonemes using cmudict gives us a repeatable method of dividing words into smaller target units than whole words. Adding AF vectors to each phoneme enables finding commonalities between disparate sounds, improving the chances of detecting and distinguishing them. In effect, AF's are additional 'hooks' to tie EMG information to. Windowing the processed data reduces the large number of EMG datapoints to a number of rows equal to the number of phonemes in the file. Training an MLPC requires a 1-to-1 correspondence between input and output rows. Discretely separating files into equal, roughly phoneme-sized bins in this manner should also accentuate cumulative spectral energy differences between phonemes. Dimensionality reduction, scaling and normalization helps regularize features of interest and balances inputs as required for reliable artificial neural net (ANN) performance. PCA helps increase signal-to-noise ratio by providing a few features capturing most of the variation of the originals, but in fewer features, making pattern-finding more likely. ANN's can easily become biased towards larger values if just a few inputs are an order of magnitude or so larger than the rest, hindering their performance, so scaling column-wise helps prevent this. Rows that have larger values will on average tend to activate ANN's more than rows with smaller values, so normalization to unit vector length row-wise helps ensure equal sample weighting.

Complex, conditional interplays of information are what ANN's excel at learning, making them potentially well suited to identifying the delicate and intricate actuations of fine motor signals. Given sufficient complexity, ANN's can reconstruct any other function. Relationships between EMG data and phonemes that produced them are noisy, conditional, and fuzzy—where ANN's can excel. Additionally, using ANN's has the advantage of vaguely emulating biological systems evolved for the same general purpose of linguistic decoding in human auditory neural circuits. Using grid search to refine our MLPC helps identify at least a local (if not global) optimum of model parameters. Not only will grid searching permutations of parameters usually result in superior results to the default model settings, but it also prevents more manual methods of tweaking parameters to improve the model.

Each component of the preprocessing and learning model pipeline uses data in a specific fashion to fulfill its function. CWT is similar to FFT, but models data as the sum of transient rather than ongoing waves. Therefore, CWT attempts to decompose the parent voltage signal into discrete, overlapping wavelets. Results of this process include amplitude values for each discrete wavelet width for each input data point. Generating phonemes with NLTK's cmudict happens simply by passing a word as a key into the dictionary, returning a list of phonemes. Each phoneme is then passed into a dictionary, built using Rasipuram as a reference, to construct AF feature vectors for each phoneme. Windowing of transformed data is accomplished through summing up squares of absolute values of amplitude for each wavelet frequency for each data vector in the window. Indexes are calculated for each window based on the number of rows of data and the number of phonemes in the file. For a given file, windows are of a fixed size, but for a different file length and phoneme number, windows may be a different size (but all equal for that file). Summation happens column-wise within a window, yielding a single row of data per window with component number equal to that of the original data vectors. Each window therefore has a single total energy spectrum associated with it. Dimensionality reduction, scaling column-wise, and normalization row-wise are then performed on all data windows. Means and standard deviations are found for each column, then old values are replaced by subtracting the mean and dividing by the standard deviation for the column. Normalization involves treating each row as a vector, finding its vector length, and dividing each value in the row by its vector length. Obtained from this process are rows of unit vector length (a value of 1) with ratios preserved. Dimensionality reduction by PCA reduces input features from 50 to 10. Specifically, PCA reduces dimensions by finding dimensional ratios, in terms of existing dimensions, which maximize variance between data points in the set. Dimensional ratios so obtained are used as a transform to map the original, higher number of dimensions onto fewer dimensions capturing at least most of the variability happening on the higher ones.

Training an MLPC, like other ANN's, involves systematically tuning input weights for each neuron in the system to reduce the result of an error function. Using X as inputs, balanced 10-dimensional vectors in this case, and y as outputs, phonemes or phoneme vectors in this case, a simple error function is used to stepwise adjust weights. Programmatic derivation is used to find the direction of weight tuning that minimizes error between what outputs should be and what they are at a given iteration. Each layer undergoes tuning in a single round, and the process of error minimization is repeated iteratively until a maximum of iterations is reached or error converges to within a certain tolerance. Grid searching enables cross validation of multiple models simultaneously using all permutations of the passed parameters (the “parameter space”). Taking advantage of multiple core CPU's, a unique MLPC is created for each point in parameter space, trained on the data, and assessed in performance on a distinct test set of data it wasn't trained on. Identical in principal to individual CV with a single model, grid search is simply a way of automating and parallelizing the process of finding optimal model parameters for a particular problem space.

Benchmark

Benchmarks assist assessment of more novel or tenuous approaches. In this case, a phoneme prediction model using only processed EMG data and no AF's as inputs will serve as the benchmark. Illustrating how well an MLPC can extract phoneme-relevant information from processed EMG, it performs in the 11-12% F1 range depending on exact parameters. Using no AF's as inputs for the benchmark demonstrates a single ANN attempting to extract all relevant information from processed EMG on its own. It is expected to fail to find some key commonalities and differences in phonetic signals on its own. Nonetheless, it should (and has) found enough information about phonemes to perform better than chance. It is intended to represent a “blank slate” model that is missing key “injected” linguistic domain knowledge (AF's in the present study). Additionally, the benchmark should highlight the utility of *specialized* ANN's as feature extractors, preprocessing pipelines, etc., and underline the notion that ANN's are great when they specialize, but struggle with being generalist. To such end, the benchmark represents a less specialized and more naive approach to learning patterns and to neural nets in particular. Ideally, the benchmark will serve as an example of the value of including AF's and AF extractors in the final solution model.

III. Methodology

Several aforementioned algorithms are used prior to model training, as part of the preprocessing pipeline. First is the wavelet transform, taking raw EMG data and returning an equal number of vectors whose components are amplitudes for discrete wavelet widths. Wavelet transformation helps extract the transient wave signal corresponding to MUAP's which generated the signal during speech muscle contractions. Second is the process of generating phonemes from words and AF vectors from phonemes. NLTK library is used to split words, and AF vectors built for each phoneme by a custom dictionary (based on Rasipuram) and pandas DataFrame building function. Cmudict and the AF builder function are called once for each word. Third is the EMG windowing process, summing the energies of each wavelet width for a span of time in the EMG data. Instantaneous wavelet amplitudes are de-signed, squared and summed according to their width, yielding a single vector for each window with a number of components equal to the number of components in each wavelet-transformed EMG vector. Fourth, the input and output dataframes for AFE models are built, and sklearn's PCA, scale, and normalize methods are used, in that order. Once AFE models are trained and tested, their outputs (predicted AF's) for all data are numeralized, one hot encoded, and then combined with the processed EMG data to serve as input for the solution model. Row labels from the output dataframe used to train AFE's serve as the basis for building a target DataFrame for the phoneme model, as these are simply the phonemes which the solution model should learn to predict.

Addressing Particularities

Employing CWT helps compensate for the relatively low voltage resolution of the EMG instrument. Inferring information from relationships between voltage values reduces the sensitivity requirements of the individual values. CWT is particularly resistant to the effects of Gaussian noise, cross-talk from control and power lines, and other stationary waves. Windowing the results of CWT to find total energy spectra also helps compensate for low voltage and temporal resolution by integrating data over time. Ideally, integration over time will help accentuate cumulative energy differences between phonemes on the 10-100ms time scale phonemes are spoken over. During the file loading phase, try/except enables automatic ignoring of missing files. In building AF vectors, only the most common phoneme variants are used to obviate issues in building vectors for phonemes that aren't listed in the academic sources employed. Further facilitating this mapping of AF's to phonemes is the casting of phoneme variants from nltk's cmudict to their simplest variant. Systematic distortions of specific wavelet widths or during an interval of recording is at least partially overcome by the scaling and norming performed on the data. Scaling and normalization have the additional benefit of regularizing ANN training and smoothing EMG data, making convergence of error during training more likely.

Implementation

Algorithms, techniques and metrics have been tailored to the project with various specific helper methods. First, using the list of words used to generate EMG data, a dictionary is built with each word as a key, and raw EMG DataFrames from all series as values. Raw EMG DataFrames are loaded directly from the EMG CSV files. Next, another dictionary with words as keys is built with DataFrames of labels. Phonemes of each word serve as row labels for the label DataFrame, and the columns are AF categories. Vectors of AF's for each phoneme, named for the phoneme they encode, are placed into a DataFrame which becomes the value of each word in the label dictionary. Raw EMG DataFrames are processed and placed into a new dictionary containing each word as a key. Processing of raw EMG involves Wavelet Transformation and then windowing, resulting in an equal number of rows of processed EMG data and phoneme labels. Processed EMG windows from the processed EMG dictionary are then placed into a single master input DataFrame ("X"); phoneme-labeled AF vector frames are placed into a master output DataFrame ("y").

Stock AF extractor models are trained and then optimized using grid search. AF columns (4) are each used as labels to train a single MLPC with stock parameters, for a total of four (4) stock MLPC models. One AF dimension per AF extractor model. AFE models take only processed EMG as inputs, yielding a single AF column ("manner", "place", "height", "vowel"). AFE models are then optimized with grid search, and their CV results compared to the CV results for the stock models to ensure relative improvement. For the benchmark model, cross validation uses only processed EMG data as input and phonemes as outputs. No AF data is used, as the benchmark acts as a "control" model for evaluating effectiveness of including AF-specific processing. In preparation for developing the solution model, the AFE model outputs are encoded and then combined with the processed EMG frames. Recall AFE's (imperfectly) predict AF's from processed EMG. Since these outputs are categorical, they are transformed to a numerical order and then one hot encoded before being combined with processed EMG vectors to serve as input for the solution model. Generated AF's (from the AFE models) are the only ones used in training and test of the solution model. Solution phoneme model using predicted AF's as inputs is then cross-validated and optimized. A stock solution model's performance is compared to a grid searched version. Grid search parameters for the solution model is similar but slightly more expansive than that of the AFE grid search parameters, with more intermediate values of "alpha" and numbers of layers (but same layer sizes).

Complications, Changes

Some unique challenges arose in the project, requiring some significant changes to methodology. Using predictions from one model's output as another model's input complicated train/test splitting somewhat. AFE models generate inputs for the solution model, so AF's generated by them as output are added to the processed EMG DataFrame before train/test splitting for CV is performed. In effect, a new hybrid input frame of predicted AF's and processed EMG is created as an input to the train/test split, the other input is simply the phoneme labels. Phoneme labels will serve as an output target or label DataFrame during solution model training and testing.

Early attempts to operationalize the pipeline involved FFT (specifically, numpy's rfft() method). FFT performs well when finding stationary waves lasting for the duration of the window passed to it. However, EMG signals are generated by MUAP's, which are non-stationary, transient waves with a more Gaussian, non-sinusoidal shape. Results from these trials indicated poor feature extraction, as parameter tuning did little to change results.

Large files with multiple words and intervening silence were used initially. An autodetector scheme involving a model trained using binary labeled data ("no vocalization" and "with vocalization") was used to help detect and automatically label phonemes sequentially within the EMG data. Usually these files were 10s or longer, with 3 repeats of a single sentence per file and approximately 2 seconds of silence separating each repeat. Naturally, this approach lead to quite imperfect labeling due to misalignment of EMG data and labels. Timing wasn't always consistent, and EMG noise made binary labeling difficult. In this arrangement, the autodetector could only be made about 90% accurate, even when using all available data to train it. Although 90% may seem acceptable, single errors in this arrangement had cumulative effects causing cumulative frame-shifting and cascade failure. Instead, a single word-per-file scheme allowed simplification of data labeling, resulting in significantly improved model accuracy. Single word-per-file made straightforward labeling possible, where one file is one word. Each file was split into a number of windows equal to the number of phonemes in the word, and each portion of the file assigned to the phoneme it should approximately contain. Ultimately, the single word-per-file approach increased the likelihood of data being labeled accurately.

Dense and Complicated Code

At various points in the development of the project, some code proved to be particularly dense or complicated. A large portion of such code is no longer in use, but some remains. Dissecting and reassembling data to form splits for the phoneme model is a more complicated portion of code that does still remain useful. AF's generated by AFE models serve as inputs to the phoneme model, but they are first generated by the AFE models from processed EMG vectors before being added to a new frame with the processed EMG vectors, one hot encoded, and then split for CV of the solution model. Splits for training and test, composed of processed EMG and AF's as input splits and phoneme targets as output splits are then used to CV the solution model.

In the current implementation, all code related to FFT windows, autodetector/autoalignment is obsolete. Older code will contain FFT windowing, but FFT was removed with the usage of CWT instead to find energy spectra bands from EMG data. Autodetector code has been removed in the recent version of the code repository, but older repositories will have it. Autoalignment (of EMG FFT windows and phoneme vector labels) was perfomred by the autodetector and was necessary for the larger multi-sentence, multi-word files. However, this technique was rendered obsolete by the new one word-per-file approach.

Refinement

Employed algorithms and techniques underwent many incremental improvements and some paradigmatic overhauls. Three major shifts were made. First, from complex files using an autodetector scheme to a single word file with simple splits. Second, from FFT to CWT in processing EMG data. Third, from solution model training with perfect AF's (and imperfect AF's from the AFE's in test) to training with AF's generated from the AFE models.

First among improvements was to move from complex EMG files to simple ones. Early files used three repeats of small to large sentences with silence interspersed between repeats. Fixed window sizes were used for all files, regardless of duration or the number of phonemes expected to appear. In that arrangement, an autodetector based on support vector classification was used to help automatically label EMG-FFT windows with phoneme vectors. On an individual basis, the autodetector was fairly accurate, with an F1 score of about 90%, but this approach led to cumulative error over the course of a file. Missing one in ten phonemes when a file is expected to have 30-50 total led to very large frame shift errors and cascade failure. Fortunately, a much simpler approach involving one word per EMG file replaced the more complex and failure-prone approach. A single word would occupy an entire file, with no intended silence. Each file was then split into evenly sized windows, the sizes of which varied according to the duration of the file and the number of phonemes expected to appear within it. While still inaccurate for several reasons, results proved to be consistently better than chance and many fold better than the more complex route.

Second, the discrete fast fourier transform (FFT) was replaced with the continuous wavelet transform (CWT). FFT was employed in the first place to provide approximate spectral energies for each window of the data. Eventually, the tendency of FFT to underestimate and misplace the energy of higher frequency non-stationary waves was noted. In particular, FFT processing yielded inconsistent results for phoneme energy spectra, with very large variances between samples for any given phoneme, even with the single word-per-file approach. An added benefit of the switch to CWT was making the processed data easier to visualize and intuitively connect to MUAP activity, with more distinct features belonging to phonemes upon simple visual inspection.

Moving from perfect AF's as inputs to the solution model during training to using imperfect AF's generated by AFE models proved to be advantageous. While imperfect AF's from the AFE models were always used to test the solution model, using them during training of the solution model as well was only seized upon later. Initially, the phoneme model just used perfect ground-truth AF's in train. Results indicated from these early trials the solution model couldn't learn to compensate when the AF input was wrong in test. Once imperfect AF's were employed in both test and training, the solution model improved markedly. Apparently, the solution candidate learned to extract supplemental information when the input AF's it was given were wrong.

Initial, Intermediate, and Final Solutions

Among the various stages of results, a variety of permutations of algorithms and techniques were employed. Documented here will be three such stages. First, using complex files, FFT, and perfect training AF's. Second, using complex files, FFT and imperfect training AF's. Third, using simple files, CWT, and imperfect train AF's.

Employing the complex file approach, FFT, and ground truth AF's in training yielded a model with about 1% F1 in testing. Accounting for its performance requires first acknowledging the cumulative error from frame-shift labeling errors with the autodetector scheme. Also, the FFT energy spectra failed to distinguish phonemes well or consistently. Using ground truth AF's as inputs for the solution model in training while using the imperfect AF's generated by AFE models in test prevented the model from learning to compensate in test. Taken together, it's almost impressive the model achieved even 1%.

A marginal improvement using imperfect AF's generated from the extractor models in both training and test of the solution model saw F1 test scores increase to the 3-5% range. Again, cumulative frame shift error from the autodetector combined with FFT's minimal distinction of phonemes likely restrained the model's performance. This time, however, imperfect AF's in both training and test likely led to the solution candidate learning to compensate somewhat for the imperfect information fed to it, and handling itself better in test. If each complex file contained 30 phonemes, and on average only the first 9 were labeled correctly, then about 30% of the data was correctly labeled, and the model was able to correctly match the equivalent of 10-20% of these in test. Most likely, the model learned to compensate somewhat for the imperfect AF's by drawing supplemental information from the processed EMG it otherwise would have ignored, as the first model appeared to.

To preview the final model, employing simple files, CWT, and imperfect AF's in training, over 13% F1 was achieved by the final model. Using a single word per file, the final pipeline used equal windows based on phoneme number and file size instead of the more complex scheme. CWT allowed the system to better identify both uniting and differentiating phoneme features. Imperfect AF's from the AFE models in training and in test also enabled the solution candidate to learn to compensate when AF information proved inaccurate.

IV. Results

Several aspects of the final model should stand to reason, including the details of the grid search, the final model parameters, and the thoroughness of cross validation. Relevant details of the grid search in determining its exhaustiveness and appropriateness include the parameters searched and their values. As regards MLPC performance, the “alpha” and “layer sizes” parameters are usually the most significant. “Alpha” changes how finely the neurons discriminate between adjacent points when making classifications. Lower values of alpha lead to finer or steeper classification gradients, potentially leading to over-fitting, while higher values cause the network to make grosser distinctions, leading to bias. Searching through layer sizes used equal layers composed entirely of either 60 or 90 neurons in each layer, with layer sizes ranging from 2 to 9. It was found that 60 and 90 neurons per layer were close to the extinction of effects on score (negative and positive, respectively) for this particular setup, and likewise for the numbers of layers chosen. More possibilities exist in this grid search for adding layers than neurons within a layer, as adding a new layer is usually more effective than adding an equal number of neurons in a single layer for data with multiple levels (or convolutions) of information.

Final model parameters include an alpha of 0.1, 7 layers of 90 neurons each, with all other values default. Alpha appears low enough to make sufficiently fine distinctions, without being so low as to encourage over-fitting. Layer sizes are large enough to find sufficient inferential supports, but not large enough to cause significant over-fitting. Further, there are enough layers to infer information about higher levels of the data, but not quite enough to be subject to significant variance in cross validation. Both parameters are near those found in other applications of MLPC's, and not particularly close to the limits of what the implementation can handle.

In validating the final solution model, a train-test split was used with 15% of examples set aside for testing. Some variation was observed between training and test, with the model often performing better in testing than in training by 1-2% in terms of F1 score. A nearly identical train-test split setup was used with each of the AFE models, and the benchmark model, except using their respective inputs and labels. In sensitivity testing, systematic forcing was used both row-wise and column-wise in separate trials, and in yet another trial, white noise was added to some data. Test scores (in terms of F1) were found to be effected by ~0% for row-wise forcing, ~-2% for column-wise forcing (when done on principal component 0), and ~-7% for white noise addition. Modifying the solution model's parameters by adding or removing a layer, increasing or decreasing layer sizes by 30-50% was found to have an effect on F1 score in the 1-2% range. Increasing or decreasing alpha by an order of magnitude tends to have an effect in the range of minus 1-3%.

Reliability of the model's results, in the larger context of the speech recognition domain, are somewhat questionable. While the model does find some EMG similarities among phonemes and learns to recall and distinguish them at least in some cases, significant improvement would be necessary for trustworthiness. When the model does mislabel phonemes, it's most likely because it cannot distinguish the necessary features. As far as the model is concerned, those features it *does* find appear too similar to one another to make a meaningful distinction in most cases. In a real-world setting, the model would miss about 5/6ths (~85%) of all individual phonemes vocalized or sub-vocalized to it. Notably, a speech model may be able to salvage this low individual phoneme performance somewhat, by using relative predicted probabilities instead of always going with the most likely one. A speech model would be able to enhance the accuracy of the system by using a-priori knowledge about how likely a phoneme is to follow another in a given context, effectively enabling re-ordering within the top few predicted phonemes from the solution model.

Quantitatively, the solution candidate did perform better than the benchmark. The benchmark achieved an F1 score of about 11.6% in test while the solution achieved about 13.4%. In detail, the benchmark achieved better scores in classes 'AA', 'EY', and 'S' than the benchmark, while the benchmark seemed to perform better with 'AH','IH', and 'L'. Considering 336 test samples and standard deviations in accuracy of 0.0801 for the benchmark and 0.0661 for the solution, the standard error for the achieved scores is +/-0.00437% and +/-0.00361%. Comparing the scores and standard errors, with a benchmark score of 11.6+0.004 and solution score of 13.4-0.003, the results do appear statistically significant. An after-error difference of about 1.8% does appear to constitute a significant difference, illustrating that at least in this case, the addition of AFE preprocessor models for an EMG to phoneme prediction model was an effective improvement over the basic non-AF enhanced approach. Whether the solution model is significantly better than the benchmark and whether it's an effective solution within the general problem domain are distinct questions, however, and answering the latter affirmatively would doubtless require refinement within several stages of the process. Presently, however, the solution can be said to be a promising improvement over more naive approaches within the problem domain, and may offer a solid foundation for developing a form of artificial telepathy.

Phoneme	Benchmark	Solution
AA	0.286	0.000
AE	0.000	0.000
AH	0.250	0.282
AO	0.000	0.000
AW	0.000	0.000
AY	0.000	0.000
B	0.000	0.000
D	0.000	0.000
EH	0.000	0.000
ER	0.000	0.000
EY	0.250	0.000
F	0.000	0.000
G	0.000	0.000
IH	0.062	0.167
IY	0.000	0.000
JH	0.000	0.000
K	0.097	0.095
L	0.000	0.111
M	0.000	0.000
N	0.000	0.000
NG	0.000	0.000
OW	0.000	0.000
P	0.000	0.000
R	0.000	0.000
S	0.175	0.142
SH	0.000	0.000
T	0.056	0.149
TH	0.000	0.000
UW	0.000	0.000
V	0.000	0.000
W	0.000	0.000
Y	0.000	0.000
Z	0.000	0.000

Illustration 3: F1 Scores for each Phoneme Category. Some categories had insufficient examples in test to be scored.

V. Conclusion

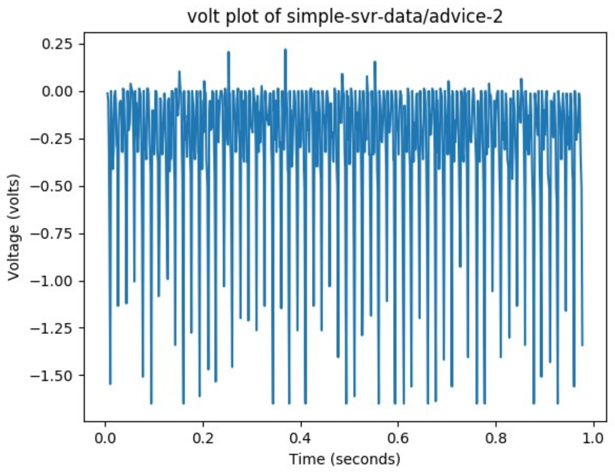


Illustration 4: Graph of raw voltage data from 'advice', series 2

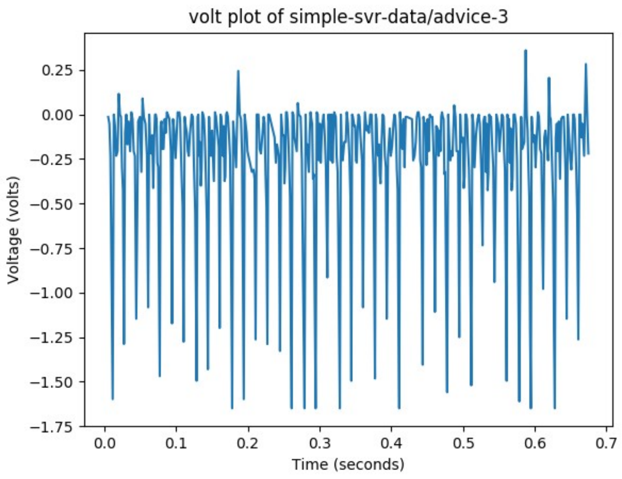


Illustration 5: Graph of raw voltage data from 'advice', series 3

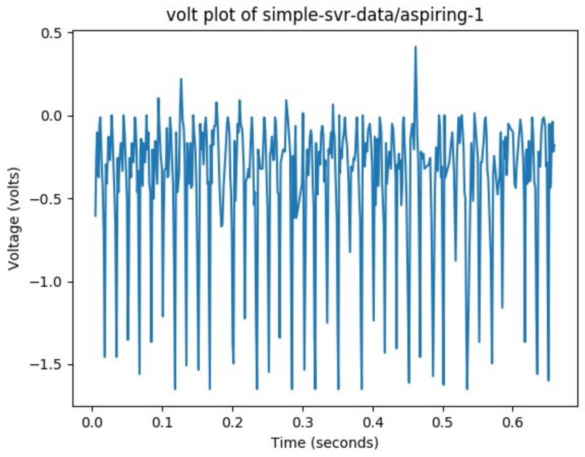


Illustration 7: Graph of raw voltage data from 'aspiring' series 1

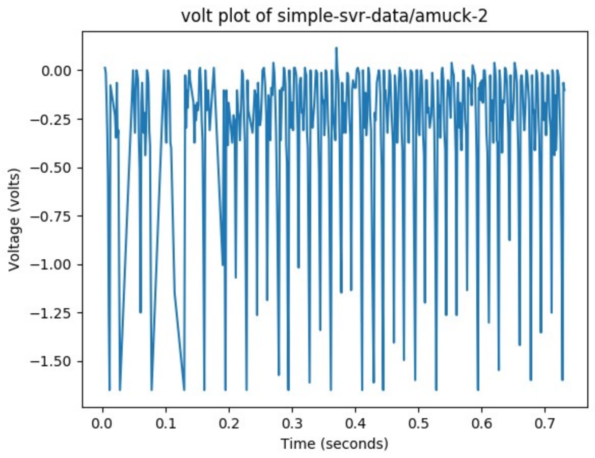


Illustration 6: Graph of raw voltage data from 'amuck' series 2

Above are visualized raw voltage graphs taken from EMG readings during vocalization. Originally time series data stored in csv format, it has been visualized as a line graph with the horizontal axis as time and the vertical axis as voltage reading. Crucially, the fairly low voltage resolution of the readings is apparent from the graph. Apparent are “pixelations” where discrete voltage levels can be seen, as only about 128 discrete levels are available from this particular setup. Despite the voltage resolution, some phonemic features appear to be visible in the raw data. Assuming that what we're seeing is a kind of phoneme correlation, they also appear to be variable in time, even for the same phoneme across different instances or words. In fact, consonants are often much shorter than vowels, and so it is expected words with more consonants would have more elongated, accentuated voltage graphs. Therefore, the timescale of variation would be expected to vary between phonemes or perhaps even between instances of the same phoneme, either within different words or for a given word during a different invocation. Using fixed windows, while convenient for training an MLPC, would be expected to lead to a loss of variation over the time frames involved with phoneme variability, leading to a loss of precision or feature resolution. If vowels are much longer than consonants, windows might end up including both or just part of a vowel, even ignoring the possibility of frame shift errors in labeling. Brief variations in spectral energy that might otherwise clearly signal a phoneme would be expected to be lost in the process of totaling energy for an entire window. Essentially, windows create big bins into which lots of small, subtle scintillations of energy are poured, making their resolution difficult or impossible. Distinguishing later which bit of energy came from which expected state proves more difficult the larger the window.

Observing the shortcomings of the energy window approach brings the focus back to why: making the data into a form usable by MLPC's. These types of neural nets usually excel at discrete classification problems, but struggle with time series and the kinds of variability that come with them. In discretizing time series data, two options present themselves:

attempt to lump lots of individual time series data points together, or attempt to further atomize or sub-divide labels into smaller pieces. In this case, the latter would have meant trying to atomize phonemes, which has no obvious way to proceed. Without a statistical, probabilistic transition model for discrete phases within phonemes, splitting them up would be an exercise in arbitrary judgement.

Now, back to the visualization. Within it, some variation is apparent, but there appears a significant amount of randomness. Higher quality EMG readouts (those using at least two (2) bytes of precision, where only one (1) was used here) appear more varied and clearly delineated between action and inaction. While a few motifs seem consistent across samples and roughly correlate to specific phonemes, much of the apparent variation is likely noise and level-jumping due to low voltage resolution. Low voltage resolution increases the difficulty of distinguishing signal and noise. Higher voltage resolution would make the difference between speaking and not more apparent. More voltage levels would be available in a higher precision ADC. In the present case, by contrast, noise or transient fluctuations can overwhelm the signal due to low voltage resolution. For some files, brief periods in the beginning and end of the file should contain silence, while the middle should contain full or sub-vocalization. Visually, however, the two are often difficult to distinguish.

From what motifs do consistently appear, those most likely corresponding to phonemes seem to vary widely in width or duration. Some of the recurrent motifs may be actual phoneme features. From those, it can be seen they have variation in duration. A single phoneme can vary in width between different samples of the same word. Additionally, it is expected that some phonemes be longer than others. Fixed energy windows struggle with probabilisticaly variable features. By compressing variations together, energy windows present some loss of reliable feature resolution. Subcomponents of phonemes that may otherwise be used as features to distinguish them are lost in the windowing method. Effectively, differing and variable phonemes feature widths are steamrolled together by the single window size employed for each unique instance of each word. As stated, no obvious way presents itself to label only portions of phonemes in time series data to facilitate MLPC usage specifically. While portions of phonemes in the EMG data have apparently distinct features, there is no obvious technique to temporally atomize phonemes into smaller pieces within a strictly MLPC framework. Another alternative would perhaps be to attempt assignment of distinct features within a phoneme to the same phoneme label, but this would likely lead to non-convergence of the MLPC, leading to inconsistent behavior at best.

Reflection

In summary, an articulatory feature extraction pipeline was investigated for detecting phonemes present within single-differential EMG data. Beginning with recording the EMG, moving to analyzing the energy spectra, training AF prediction models, automatically labeling AF's within the processed data, training phoneme predictors, and ultimately yielding phoneme likelihoods for each data window. Recording the EMG data began with preparing the word list. Next, recording began by typing the word, pressing 'enter', full or sub-vocalization, and stopping the recording—repeated until all words have samples recorded. Analyzing energy spectra was accomplished with CWT upon each EMG file to help distinguish phonemes. Training AFE models to automatically label data and having them do so followed. Words were broken into phonemes, AF vectors were generated, and DataFrames of labels were built before using them along with processed EMG data to train the AFE models. A benchmark was developed using the processed EMG and phoneme labels only. Training the solution model required combining processed EMG with predicted AF's from the AFE models. Building the input and output DataFrames for this stage preceded train-test splitting, and then actual training.

Both interesting and difficult was the task of recording good EMG data sets. EMG has been an active area of research for over 50 years. Hardware involved can range from a simple single-byte ADC to a battery of $100k+ equipment with arrays of specialized amplifiers and filers. EMG can be performed purely from the surface of the body with dermal electrodes, or with various invasive devices, usually based on conductive needles. Electrodes themselves can vary wildly in quality and be monopolar or multipolar. During recording, interference in the sensor lines and from the electrical environment can greatly impact results. Recording procedures themselves can vary from the atomic (say, single word-per-file, or simple actions) to the complex (compound actions, whole activities). With subvocalization in paritcular, an individual may be a “loud” sub-vocalizer one day, strongly activating the muscles of speech while remaining silent, while being much “quieter” the next. EMG presents particularities with preprocessing that pose an interesting challenge as well. While FFT is the simplest and usually the most common approach to extracting MUAP information, it only works well when activity is sustained for the full width of an observational window. Speech, on the other hand, is highly variable in duration and definitely transient, necessitating the use of a non-stationary wave extraction technique like CWT. Subsequently, careful selection of a base waveform and appropriate wavelet widths needs to be undertaken.

Answering whether the solution is appropriate for general usage is multi-fold. Preprocessing used here for the solution pipeline is the best among the tried alternatives, and is either close to or at the state of the art. Specifically, the software filtering of the EMG signal involving CWT, PCA, scaling, and normalization should constitute state of the art.

However, using dedicated AF neural nets for EMG is a newer and less established approach. “Less established” could translate to “disruptive” if it can be made to work well with some other changes. Neural net-based extraction of articulatory features in automatic audio speech recognition is used by Google, for instance. Primarily, improved voltage and time resolution of the EMG measurement appears to be the main hurdle. Currently, the solution uses something like the equivalent of a 16x8 pixel monochrome video where pixels can only be on or off. It is difficult to imagine using such a data feed for effective automatic sign language recognition, but that's closely analogous to the situation in the present study. Luckily, these facts taken together mean the current solution should be near the lower end of the s-curve for model performance versus data quality. Performance could stand to improve several fold, and with much better data to use, the solution probably could be made to work well enough for general usage. To help compensate for the low individual phoneme accuracy, word and n-gram models could be applied as another layer of refinement. In such an arrangement, even a slight improvement to phoneme recognition (compared to where it is now, in this solution) would likely translate to an even larger impact on word or phrase accuracy. Word and n-gram models are by design meant to handle and improve upon low component-wise recognition. Realistically, however, wrapping the current solution in a word model would likely still prove too poor for general use. Generally, the approach used here with better data and a few other tweaks could probably be ready for prime time.

Improvement

Of all possible avenues of improvement, the most promising is via the estimator at the core of the solution. In lieu of an MLPC, a hierarchical hidden markov model (HHMM) would likely boost the performance of a sub-vocal recognition pipeline several fold. Using single words per file, with hierarchical states at the phoneme, word and phrase levels would likely present a superior improvement. At the word level, the number of states would be the number of phonemes in the word. On the phoneme level, the number of states would be experimentally determined in CV to get just the right number of states and their transition probabilities for each type of phoneme. On the n-gram and phrasal level, standard libraries could be used to build n-gram probabilities. Phoneme-level recognition could be further enhanced using neural nets as AF extractors in a preprocessing pipeline, after appropriate state numbers and transitions are determined experimentally in a process of bootstrapping.

All improvements mentioned, taken together, would probably improve the solution to a widely useful state. Using a 16 bit rather than 8 bit ADC, with at least 800Hz sampling rate, and a programmable amplifier would take resolvable voltage levels from 256 (128ish in practice) to 65,536. In turn, MUAP detection sensitivity or precision should improve by at least an order of magnitude. CWT could be employed using a wave form more true to the underlying action potential of neurons. Currently, a Ricker wave is used, which is the second derivative of the Gaussian function, whereas neuronal action potentials are more asymmetric and abrupt in their shifts of value. EMG and transformation improvements alone would likely increase accuracy of an MLPC based approach like the one here by a few fold. Individual phoneme recognition would be improved through better wavelet extraction and higher fidelity signals. Ultimately, combining these with an HHMM approach, however, would likely greatly improve performance. Adaptation to variable feature scales and using priors to constrain predicted likelihoods by means of an HHMM would itself likely improve the flexibility of both AF and phoneme representation and extraction. Taken together, with improved measurement, transformation, and probabilistic modeling, a formidable improvement to the current solution seems possible and likely.

References

[1] "NASA -", Nasa.gov, 2004. [Online]. Available: https://www.nasa.gov/centers/ames/news/releases/2004/04_18AR.html. [Accessed: 18- Mar- 2017].

[2] S. Jou and T. Schultz, "EARS: Electromyographical Automatic Recognition of Speech.", BIOSIGNALS, vol. 1, pp. 3-12, 2008. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.154.6348. [Accessed: 18- Mar-2017].

[3] M. B. I. Reaz, M.S. Hussain and F. Mohd-Yasin, “Techniques of EMG signal analysis: detection, processing, classification and applications.”, Biol. Proceed. Online vol. 8, pp. 11-35, 2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1455479/ [Accessed: 18-June-2017].

[4] R. Rasipuram and M. Magimai-Doss, “Articulatory feature based continuous speech recognition using probabilistic lexical modeling.”, Comput. Speech Lang. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.csl.2015.04.003 [Accessed: 20-May-2017].

[5] B. Coe, "bwc126/MLND-Subvocal", *GitHub*, 2017. [Online]. Available: https://github.com/bwc126/MLND-Subvocal. [Accessed: 18- Mar- 2017].

Appendix A

Map of Words to Phonemes	
Word	Phonemes
absorbing	['AH' 'B' 'Z' 'AO' 'R' 'B' 'IH' 'NG']
actually	['AE' 'K' 'CH' 'AH' 'W' 'AH' 'L' 'IY']
addicted	['AH' 'D' 'IH' 'K' 'T' 'AH' 'D']
advice	['AE' 'D' 'V' 'AY' 'S']
aftermath	['AE' 'F' 'T' 'ER' 'M' 'AE' 'TH']
amazing	['AH' 'M' 'EY' 'Z' 'IH' 'NG']
amuck	['AH' 'M' 'AH' 'K']
animated	['AE' 'N' 'AH' 'M' 'EY' 'T' 'AH' 'D']
aspiring	['AH' 'S' 'P' 'AY' 'R' 'IH' 'NG']
barbarous	['B' 'AA' 'R' 'B' 'ER' 'AH' 'S']
black	['B' 'L' 'AE' 'K']
board	['B' 'AO' 'R' 'D']
bored	['B' 'AO' 'R' 'D']
bouncy	['B' 'AW' 'N' 'S' 'IY']
bow	['B' 'AW']
bright	['B' 'R' 'AY' 'T']
cakes	['K' 'EY' 'K' 'S']
caption	['K' 'AE' 'P' 'SH' 'AH' 'N']
certain	['S' 'ER' 'T' 'AH' 'N']
check	['CH' 'EH' 'K']
chop	['CH' 'AA' 'P']
clean	['K' 'L' 'IY' 'N']
clumsy	['K' 'L' 'AH' 'M' 'Z' 'IY']
cluttered	['K' 'L' 'AH' 'T' 'ER' 'D']
coal	['K' 'OW' 'L']
combative	['K' 'AH' 'M' 'B' 'AE' 'T' 'IH' 'V']
complete	['K' 'AH' 'M' 'P' 'L' 'IY' 'T']
confuse	['K' 'AH' 'N' 'F' 'Y' 'UW' 'Z']
copy	['K' 'AA' 'P' 'IY']
country	['K' 'AH' 'N' 'T' 'R' 'IY']
courageous	['K' 'ER' 'EY' 'JH' 'AH' 'S']
cracker	['K' 'R' 'AE' 'K' 'ER']
curious	['K' 'Y' 'UH' 'R' 'IY' 'AH' 'S']
dam	['D' 'AE' 'M']
dance	['D' 'AE' 'N' 'S']
defective	['D' 'IH' 'F' 'EH' 'K' 'T' 'IH' 'V']
direful	[]
division	['D' 'IH' 'V' 'IH' 'ZH' 'AH' 'N']
doubtful	['D' 'AW' 'T' 'F' 'AH' 'L']
ducks	['D' 'AH' 'K' 'S']
dust	['D' 'AH' 'S' 'T']
dusty	['D' 'AH' 'S' 'T' 'IY']
efficient	['IH' 'F' 'IH' 'SH' 'AH' 'N' 'T']
eminent	['EH' 'M' 'AH' 'N' 'AH' 'N' 'T']
enthusiastic	['IH' 'N' 'TH' 'UW' 'Z' 'IY' 'AE' 'S' 'T' 'IH' 'K']
equable	['EH' 'K' 'W' 'AH' 'B' 'AH' 'L']
expand	['IH' 'K' 'S' 'P' 'AE' 'N' 'D']
fancy	['F' 'AE' 'N' 'S' 'IY']
faulty	['F' 'AO' 'L' 'T' 'IY']
fax	['F' 'AE' 'K' 'S']
feeling	['F' 'IY' 'L' 'IH' 'NG']
flap	['F' 'L' 'AE' 'P']
flowers	['F' 'L' 'AW' 'ER' 'Z']
frogs	['F' 'R' 'AA' 'G' 'Z']
fry	['F' 'R' 'AY']
geese	['G' 'IY' 'S']
gold	['G' 'OW' 'L' 'D']
haircut	['HH' 'EH' 'R' 'K' 'AH' 'T']
hang	['HH' 'AE' 'NG']
hobbies	['HH' 'AA' 'B' 'IY' 'Z']
hop	['HH' 'AA' 'P']
icicle	['AY' 'S' 'IH' 'K' 'AH' 'L']
imperfect	['IH' 'M' 'P' 'ER' 'F' 'IH' 'K' 'T']
increase	['IH' 'N' 'K' 'R' 'IY' 'S']
innocent	['IH' 'N' 'AH' 'S' 'AH' 'N' 'T']
interest	['IH' 'N' 'T' 'R' 'AH' 'S' 'T']
invent	['IH' 'N' 'V' 'EH' 'N' 'T']
jog	['JH' 'AA' 'G']
kick	['K' 'IH' 'K']
kitty	['K' 'IH' 'T' 'IY']
knock	['N' 'AA' 'K']
languid	['L' 'AE' 'NG' 'G' 'W' 'AH' 'D']
license	['L' 'AY' 'S' 'AH' 'N' 'S']
living	['L' 'IH' 'V' 'IH' 'NG']
load	['L' 'OW' 'D']
loose	['L' 'UW' 'S']
loving	['L' 'AH' 'V' 'IH' 'NG']
low	['L' 'OW']
magical	['M' 'AE' 'JH' 'IH' 'K' 'AH' 'L']
march	['M' 'AA' 'R' 'CH']
marry	['M' 'EH' 'R' 'IY']
measly	['M' 'IY' 'Z' 'L' 'IY']
mine	['M' 'AY' 'N']
mixed	['M' 'IH' 'K' 'S' 'T']
money	['M' 'AH' 'N' 'IY']
motion	['M' 'OW' 'SH' 'AH' 'N']
nondescript	['N' 'AA' 'N' 'D' 'IH' 'S' 'K' 'R' 'IH' 'P' 'T']
number	['N' 'AH' 'M' 'B' 'ER']
outstanding	['AW' 'T' 'S' 'T' 'AE' 'N' 'D' 'IH' 'NG']
paddle	['P' 'AE' 'D' 'AH' 'L']
pan	['P' 'AE' 'N']
person	['P' 'ER' 'S' 'AH' 'N']
planes	['P' 'L' 'EY' 'N' 'Z']
poised	['P' 'OY' 'Z' 'D']
precede	['P' 'R' 'IH' 'S' 'IY' 'D']
pretty	['P' 'R' 'IH' 'T' 'IY']
puncture	['P' 'AH' 'NG' 'K' 'CH' 'ER']
push	['P' 'UH' 'SH']
quaint	['K' 'W' 'EY' 'N' 'T']
quartz	['K' 'W' 'AO' 'R' 'T' 'S']
quince	['K' 'W' 'IH' 'N' 'S']
race	['R' 'EY' 'S']
reading	['R' 'EH' 'D' 'IH' 'NG']
real	['R' 'IY' 'L']
regret	['R' 'AH' 'G' 'R' 'EH' 'T']
rely	['R' 'IH' 'L' 'AY']
river	['R' 'IH' 'V' 'ER']
rod	['R' 'AA' 'D']
round	['R' 'AW' 'N' 'D']
rub	['R' 'AH' 'B']
save	['S' 'EY' 'V']
separate	['S' 'EH' 'P' 'ER' 'EY' 'T']
shoe	['SH' 'UW']
signal	['S' 'IH' 'G' 'N' 'AH' 'L']
sound	['S' 'AW' 'N' 'D']
spare	['S' 'P' 'EH' 'R']
spiritual	['S' 'P' 'IH' 'R' 'IH' 'CH' 'AH' 'W' 'AH' 'L']
squirrel	['S' 'K' 'W' 'ER' 'AH' 'L']
stereotyped	['S' 'T' 'EH' 'R' 'IY' 'AH' 'T' 'AY' 'P' 'T']
stingy	['S' 'T' 'IH' 'N' 'JH' 'IY']
stomach	['S' 'T' 'AH' 'M' 'AH' 'K']
straw	['S' 'T' 'R' 'AO']
sun	['S' 'AH' 'N']
superb	['S' 'UH' 'P' 'ER' 'B']
swim	['S' 'W' 'IH' 'M']
talented	['T' 'AE' 'L' 'AH' 'N' 'T' 'AH' 'D']
third	['TH' 'ER' 'D']
thirsty	['TH' 'ER' 'S' 'T' 'IY']
throne	['TH' 'R' 'OW' 'N']
tightfisted	['T' 'AY' 'T' 'F' 'IH' 'S' 'T' 'IH' 'D']
tired	['T' 'AY' 'ER' 'D']
tow	['T' 'OW']
town	['T' 'AW' 'N']
treatment	['T' 'R' 'IY' 'T' 'M' 'AH' 'N' 'T']
trick	['T' 'R' 'IH' 'K']
type	['T' 'AY' 'P']
ugliest	['AH' 'G' 'L' 'IY' 'AH' 'S' 'T']
uncle	['AH' 'NG' 'K' 'AH' 'L']
unsuitable	['AH' 'N' 'S' 'UW' 'T' 'AH' 'B' 'AH' 'L']
utter	['AH' 'T' 'ER']
vague	['V' 'EY' 'G']
various	['V' 'EH' 'R' 'IY' 'AH' 'S']
vest	['V' 'EH' 'S' 'T']
wait	['W' 'EY' 'T']
wander	['W' 'AA' 'N' 'D' 'ER']
weather	['W' 'EH' 'DH' 'ER']
whisper	['W' 'IH' 'S' 'P' 'ER']
wrathful	[]

Appendix B

Phonemes Mapped to Articulatory Features				
Phoneme	Manner	Place	Height	Vowel
' '	['silent'	'silent'	'silent'	'silent']
'AA'	['vowel'	'back'	'low'	'yes']
'AE'	['vowel'	'mid-front'	'low'	'yes']
'AH'	['vowel'	'mid'	'mid'	'yes']
'AO'	['vowel'	'back'	'mid-low'	'yes']
'AW'	['vowel'	'mid-front'	'low'	'yes']
'AY'	['vowel'	'back'	'low'	'yes']
'B'	['voiced-stop'	'labial'	'max'	'no']
'CH'	['stop'	'front'	'max'	'no']
'D'	['voiced-stop'	'alveolar'	'max'	'no']
'DH'	['voiced-fricative'	'dental'	'max'	'no']
'EH'	['vowel'	'mid-front'	'mid'	'yes']
'ER'	['vowel'	'mid'	'mid'	'yes']
'EY'	['vowel'	'front'	'mid-high'	'yes']
'F'	['fricative'	'labial'	'max'	'no']
'G'	['voiced-stop'	'dorsal'	'max'	'no']
'HH'	['aspirated'	'unknown'	'max'	'no']
'IH'	['vowel'	'mid-front'	'high'	'yes']
'IY'	['vowel'	'front'	'very high'	'yes']
'JH'	['voiced-stop'	'front'	'max'	'no']
'K'	['stop'	'dorsal'	'max'	'no']
'L'	['approximant'	'lateral'	'very high'	'no']
'M'	['nasal'	'labial'	'max'	'no']
'N'	['nasal'	'alveolar'	'max'	'no']
'NG'	['nasal'	'dorsal'	'max'	'no']
'OW'	['vowel'	'back'	'mid'	'yes']
'OY'	['vowel'	'back'	'mid-low'	'yes']
'P'	['stop'	'labial'	'max'	'no']
'R'	['approximant'	'retroflex'	'mid-low'	'no']
'S'	['fricative'	'alveolar'	'max'	'no']
'SH'	['fricative'	'front'	'max'	'no']
'T'	['stop'	'alveolar'	'max'	'no']
'TH'	['fricative'	'dental'	'max'	'no']
'UH'	['vowel'	'mid-back'	'high'	'yes']
'UW'	['vowel'	'back'	'very-high'	'yes']
'V'	['voiced-fricative'	'labial'	'max'	'no']
'W'	['approximant'	'back'	'very-high'	'no']
'Y'	['approximant'	'front'	'very-high'	'no']
'Z'	['voiced-fricative'	'alveolar'	'max'	'no']
'ZH'	['voiced-fricative'	'front'	'max'	'no']