CS221 Project Progress Report

Modeling Infant Statistical Learning in Lexical Acquisition through Machine Learning

Jihee Hwang (jiheeh), Krishan Kumar (krishank), Eric Ehizokhale (eokhale)

A. Model and preliminary algorithm:

Our statistical learning model seeks to effectively segment fluent speech into words, as infants do in lexical acquisition, which is difficult as there are not often clear boundaries between words in natural speech. However we know that that infants can statistically find patterns of certain phonemes that repeatedly occur across different utterances to find out the individual words making up the lexicon.

These statistical regularities can be represented using transitional probabilities from one syllable of a word to the next. The transitional probabilities will be higher when two sounds occur after one another within words, and lower between the words themselves. So for instance, given the two words "dagger tattoo," the transitional probabilities between *da* - *gger* and *ta* - *ttoo* will be higher than the transitional probability between *ger* and *ta*. Analyzing these probabilities from a set of data will allow us to determine a threshold probability that indicates a break between words.

To determine the break between words in a given sentence, the following steps and algorithm will be taken. Given an example data set, an audio file of someone speaking the words: "table dagger fazer dagger table fazer":

1) Take a given audio file and insert breakpoints in the recorded speech after every syllable, using an onset detection function provided by the Librosa python library. The algorithm determines the peaks of an onset strength envelope, i.e. the highest point of a waveform. These peaks will often be the beginnings of syllables, as these points are the parts of the waveform with the greatest intensity (we emphasize the beginnings of syllables when speaking, especially those starting with consonants). We have tested out various onset strength envelope constants in order for the function to be the most accurate at finding breakpoints between syllables.

2) Splice the audio file into segments between every breakpoint. These will roughly be the different syllables.

3) Use k-means clustering algorithm to generate syllable clusters in order to generalize the different syllables. This is necessary to identify when the same syllable occurs multiple

times across an audio file. In the given example, for instance, clustering would allow us to identify the “ta” syllable in the first occurrence of “table” as the same “ta” syllable in the second occurrence of “table.” Currently, we are using two features for the k-means audio clustering.

One is the average zero crossing rate, which refers to the number of time an audio signal crosses through zero. This value would be highly correlated to the frequency of the audio segment. Moreover, the feature could also help distinguish between different phoneme properties, such as between voiced ([d]) / unvoiced([t]).

The second feature is the short-term energy, which is essentially the norm of the audio segment interpreted as a vector. An amplitude could be a good way of distinguishing speech from silence, and also between relatively silent and loud phonemes.

4) Create a map where the keys are tuples of two adjacent syllables. This will be used to calculate the transitional probabilities between syllables. Iterate through the whole data input of audio files, and count the number of times a syllable pair appears. For example, for the given example data, the map values would look like:
Map: (ta, ble) = 2, (ble, da) = 1, (da, gger) = 2, (gger, fa) = 1, (fa, zer) = 2, (zer, da) = 1, (gger,ta) = 1, (ble, fa) = 1

5) Calculate the transitional probabilities between syllables by comparing the counts for each key in the map. This equals the count of each tuple pair divided by the total number of times all the tuples occur. Again, for the given pairs of syllables presented in step (4), the transitional probabilities would be each count divided by 11, giving transition probabilities of 18%, 9%, 18%, 9%, 18%, 9%, 9%, 9% respectively. The resulting map would look like this:
Map: (ta, ble) = 0.18, (ble, da) = 0.09, (da, gger) = 0.18, (gger, fa) = 0.09, (fa, zer) = 0.18, (zer, da) = 0.09, (gger, ta) = 0.09, (ble, fa) = 0.09

   We have previously discussed that syllable pairs with high transitional probabilities are likely to be a part of the same word, as they would always be uttered together. To find breaks between words, we would pick a constant threshold, which will then indicate that any pairs with a transitional probability below that would indicate a word break. For example, if we decide that 0.10 would be the threshold, then any time (ble, da), (zer, da), (gger, fa), (gger, ta) and (ble, fa) appears would breaks between words. For now, choosing the right threshold would involve human judgement.
   This would reconstruct our sentence as:

   Table dagger fazer dagger table fazer

6) The audio is then segmented once more, this time between breakpoints determined as word breaks by step (5). This would produce a set of word candidates; however, our goal of this project is to generate a *lexicon* from a given audio data set, and not all word segments that were ever uttered. Not only would some word candidates not be an actual word, but there would be many words that would reoccur across the audio files as well. Thus, another step of clustering will be necessary to generalize multiple utterances of the same word into one. We would repeat step (3) for this.

7) Once we're done, hopefully the k-means would have generated a lexicon that was used for the set of audio files. In the future, when we are planning to feed the algorithm with a much larger set of audio files and different words, we would print out the top n number of audio segments that we are most confident constitutes a word. An exact method to do this is yet undetermined.

B. Initial results:

Baseline: The baseline algorithm uses silence detection to note differences between words. This performs mediocrely, as normal spoken language doesn't have notable pauses between words. Depending on the parameters set for silence detection, we can either identify roughly 90% of word boundaries but get a slew of false positives, or we can be more conservative and require longer silence between words, which gets fewer false positives but closer to 60% of words identified on a monosyllabic dataset. The false positive problem is exacerbated for multisyllabic audio.

Our algorithm: As it stands, onset detection is better at detecting syllables than silence is at detecting words — there are fewer false positives with onset detection with more syllables captured. If we control for syllable detection and clustering issues by recording clear utterances of each syllable in order, then our pipeline is more effective than the baseline in identifying words. Some mistakes may still occur which is dependent on the data. For example, in the sentence "table dagger tazer dagger table tazer" (note that this example is different than the one presented in section A), the occurrences of [dagger] and [ggerta] are equal, so they're both just as probable to be identified as words. This problem becomes less significant with longer audio files and more data.

However, if we process audio with naturally spoken language, we are currently unable to perform better than the baseline. Normal spoken English doesn't enunciate each syllable super clearly, so our syllable detection misses more syllables than when we create clearly spoken audio

files. More critically, the k-means clustering of our syllables is not adequate enough. Because we have a small number of features we're using for k-means, there is nontrivial overlap between clusters, so some syllables are incorrectly labelled and thus the probability table is compromised. We're working to improve both syllable detection and feature selection of our audio to make k-means clusters as distinct as possible.

C. Challenges:

Features: We need high dimensional audio features to pass into our clustering algorithm. Why? Even with a small example (3 distinct syllables and 9 total utterances of those syllables) the clusters are close together and our current iteration of k-means might incorrectly group some syllables depending on the enunciation.

Choice of clustering algorithm: We've so far used k-means as our algorithm to (a) group utterances of syllables together to label syllable occurrences by their cluster and to (b) group similar words at the end of the algorithm to find the "strongest" identified words in the dataset. We have chosen k-means because we've been testing on short audio files where we can count the correct number of clusters, but for word detection in a long file, such as a children's audiobook, it would be hard to identify the correct value of k because we could only estimate the expected number of syllables. To fix this issue, we may use affinity propagation when looking at long audio files. AP is a clustering algorithm similar to k-means with the caveat that you don't have to determine the number of clusters to AP before running the algorithm.

Syllable detection: We're currently using the librosa onset detection feature to find syllables in audio, which works by setting a delta threshold for the change in volume between two points of audio, and marking parts where there is a noticeable change as a syllable. In our own generated test audio files, in which we recorded small passages of spoken text, this gives imperfect but "good enough" results — most syllables are found with only a few false positives. Additional refining has to be done to make this syllable detection work for general audio files, like children's books or podcasts. Speakers don't always enunciate their words clearly for them to be detected in this method, and certain words lend themselves to be more easily detected in this method. For example: the second syllable in *da - gger* is a soft g, and the onset detection function sometimes misses utterances of this second syllable. But both syllables in *ta - ttoo* are hard and clear, so the onset detection has been able to consistently identify both syllables.