CS221 Final Report

Modeling Infant Statistical Learning in Lexical Acquisition through Machine Learning

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

Introduction:

Infant language acquisition is an efficient process that has been studied by linguists and psychologists for decades. While we still don't fully understand yet how an infant can learn a language simply by listening to the environment around them, the most prominent theory proposed to explain the phenomenon so far is the statistical learning model.

The statistical learning model for infant language acquisition theorizes that children obtain lexical, grammatical and even phonological knowledge about a certain language by extracting statistical regularities from any input stream of words. The most well-observed one out of those is the learning in lexical acquisition; while spoken language does not have clear boundaries between words, infants can statistically find patterns of certain phonemes that repeatedly occur across different utterances to figure out the individual words making up the lexicon. For example, consider the given two grammatically acceptable sentences: *This is a dog* and *This was a lemon*. While the two sets of three morphemes *is-a-dog, was-a-lemon* would be pronounced without any breaks in the middle, the child can *statistically* realize that *a* is the repeatedly occurring component, and thus conclude that it is a single word . For this project, we aim to replicate such lexical statistical learning given a set of speech audio inputs by incorporating AI principles learned in class.

Baseline and oracle performance:

The baseline algorithm to detect words involves looking for breaks, or short pauses, in speech to separate words. The break is defined as local minima in volume.

With a simple dataset of 24 sentences spoken slowly, each sentence being a permutation of the words "I see Spot run", the baseline is able to identify the words "I", "see", and "spot" with acceptable accuracy (it struggles with the word run and separates the word in the middle).

However, spoken language does not contain noticeable breaks between words, so problems arise with the baseline when the words in the spoken audio is multisyllabic and spoken at a normal speed. One example is when the baseline tries parsing the sentence, "*For even quicker answers, all the online resources above are searchable*." With a pause length of 20 ms, the algorithm doesn't successfully find each individual word, instead identifying streams of

audio like, "*For even quick*" as one word. With a smaller pause length of 5 ms, the algorithm incorrectly identifies 50 words in the passage because it separates words too often.

Oracle algorithm performance equates to a human listener fluent in the language identifying spoken words in any data set. This results in a nearly 100% accuracy in identifying words.

Algorithm overview:

Our statistical learning model seeks to effectively segment fluent speech into words, as infants do in lexical acquisition. 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 the most likely set of phonemes to occur together in a given data set.

Thus, our algorithm can be broken up into three major parts — (1) detecting syllables in audio, (2) grouping syllables together, and (3) finding the likelihood of certain syllable combinations happening in audio.

Detecting Syllables in Audio:

Infants exposed to language have to identify the set of sounds (phonetic segments) relevant to that language and, eventually, map them onto the phonological categories for that language[1]. These would eventually become the building blocks of each lexicon; thus, it is important that in modeling language acquisition, we also segment an audio stream into a meaningful collection of phonemes, which can be used later to define individual words.

This is done by first inserting 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 highly correlate with beginnings of syllables, as these points are the parts of the waveform with the greatest intensity (we generally tend to emphasize the beginnings of syllables when speaking, especially those starting with consonants). We have tested out various onset strength envelope constants and peak detection thresholds in order for the onset detection to be the most accurate at finding breakpoints between syllables. Afterwards, we split the file into segments between every breakpoint.

[1] Clark, Eve V. *First language acquisition*. Cambridge: Cambridge U Press, 2009.

Grouping Syllables Together:

To determine the phonological structure of a language, it is important to be able to generalize each instance of an uttered sound so that an utterance produced by different speakers in different timbre and accents can be perceived together as a single phonological category. For example, if a speaker says the word "dagger" multiple times in a speech, we want each utterance of the syllable "dag" to be clustered together despite small differences in how the word was spoken (enunciation, volume, etc). However, such categories are language-specific -- a two set of utterances could be categorized as the same syllable in one, and not in the other. Thus, we will be clustering the phonological components detected in the first step, to generalize a set of categories for this specific language.

We use the affinity propagation (AP) clustering algorithm to generate syllable clusters in order to generalize the different phonemes. AP is a clustering algorithm similar to k-means, but it has the added benefit that you don't have to determine the number of clusters to AP before running the algorithm. Not having a designated number of clusters is necessary for general cases when we would not know the number of syllables in an audio stream. Knowing the number of phonological categories beforehand is clearly not an option for infants, as well.

To cluster the syllables accurately, designing the right features is crucial. Through different experimentations, we have seen that the Mel Frequency Cepstral Coefficient, or MFCC, is a very robust feature for audio classification as it correlates with the short time power spectrum of audio. This better identifies linguistic content, like the shape of the vocal tract, tongue and teeth placement, etc., when audio is uttered. We compute a constant amount (13) of MFCCs per syllable and pass in these values to our AP algorithm. Performance of MFCCs in clustering syllables is significantly better than performance of some of our previous features, such as zero-crossing rate (correlated to frequency) or short-term energy (norm of audio vector, correlates with amplitude).

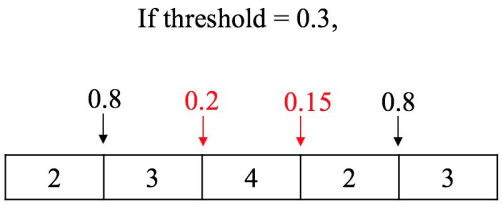
As a small example, in the *jihee_long* dataset, a 2-minute self-recorded dataset comprised of 6 bi-syllabic words {*guitar, tattoo, table, dagger, panda, beaker},* there are 11 distinct syllables. When clustering on MFCC features, we get clustering accuracy of 90-plus percent, with only small errors with similar sounding syllables *-ger and -ker*. Clustering on only frequency and amplitude features gave near random results. Two segments of audio can be spoken at similar pitches and volumes throughout, but pitch and volume alone don't provide clear characteristics of the linguistic content of spoken audio.

Calculating word probabilities:

Breakpoint Method:

Once we've identified syllables in an audio file and correctly grouped them, we then calculate the probabilities of words given the occurrences of syllables in succession. This data will help us reconstruct the final word candidates from a given audio stream. During our project, we went through two different models to generate our final lexicon.

In our milestone, we first created a map where the keys are tuples of two adjacent syllables, then calculated the transitional probabilities between syllables by comparing the counts of each key in our map. To identify a syllable pattern as a word, we set an arbitrary threshold for an audio file, then split the file on all syllable transitions that had a probability lower than that threshold. This would work in cases when for instance, the probability of syllables (da, gger) happening in succession would be much higher than the probability of syllables (gger, the) coming together, which would result in a segmentation between (gger, the) rather than within the word *dagger*. Putting this into a syllable label array, assuming that the syllables have been correctly categorized, the method would look like the following:



Where {(23), (4), (2)} would be found to be the words in this audio file. Then, from these word segmentations of the raw audio file, we run the clustering algorithm again from step 2 to cluster different instances of the spoken word. We believed that this would lead to a generalized set of lexicon where each cluster would contain multiple spoken instances of that word.

This methodology proved effective for the *jihee_long* dataset, a 2-minute self-recorded dataset comprised of 6 bi-syllabic words {*guitar, tattoo, table, dagger, panda, beaker* }. The algorithm successfully identified 4 different words in the lexicon, and while struggling to cluster some of the words, were able to cluster 3 different words together in the same cluster. However, this poses three problems: first, the threshold constant that determines where the breakpoints are tends to be a "magic number" and it is difficult to deterministically decide upon the most optimal threshold. Second, the successful identification was likely only possible because all the words that made up the lexicon in the data were bi-syllabic, thus being able to avoid any errors resulting from using bi-syllabic translational probabilities for uni-syllabic or tri-syllabic words. The second problem that this method poses is that it is impossible to generate a lexicon in the order of probability -- in other words, in the case where the dataset is longer and the segmentation is less accurate, such as audio files with a large vocabulary with less permutations of the same word, the clustering algorithm will also fail to output a consistent set of lexicon.

Uniform Cost Search Method:

To address the above issues, we made two modifications to our existing algorithm. First, we are now counting the co-occurrences of not only tuple of syllables, but more than two combinations of them. We first set a max number of syllables that we assume to occur in a single word, which in our implementation was 4. Even the most complicated vocabularies should have at the very most 6 syllables. This works like the following:

{ 2, 5, 2, 5, 8, 1, 2, 5, 2, 6, 2, 4, … }

We take in an integer array of segment labels, and assuming that the syllable segmentation is correct, each number will correspond to each phonological category. Now we will start filling out our occurrence count dictionary. For the first iteration, we will count 1 for each of the following tuples: (2), (2,5), (2,5,2), and (2,5,2,5). For the second iteration, we will count (5), (5,2), (5,2,5), (5,2,5,8), respectively. Notice that in the third iteration however, our occurrence count becomes:

{(2): 2, (2,5): 2, (2,5,2): 1, (2,5,2,5): 1, (5): 1, (5,2): 1, (5,2,5): 1, (5,2,5,8):1…}

Where we can notice that there are multiple occurrences of the syllable (2) and (2,5), signifying that these are more likely to be combined to form a word. Now, after we iterate through the audio file, we will end up with a list of syllable combinations that are most likely to occur in this audio stream.

However, this data will not be used as it is but instead will be inputted as a cost function to the Uniform Cost Search we will be running. Because it must be a cost, not a reward, combinations more likely to be a word must have a lower cost; therefore, we process this value such that each element becomes =

$$\frac{1}{((\frac{occurrences[key]}{sum(occurrences)}) * log(len(key) + 0.2))}$$

We are first dividing the value of each key by the total number of occurrences counted, to change the count values to a probability value. Then, we're applying a log on it in order to give more weights to longer tuples; for example, if (2,3,4) has the translational probability value 0.6 and (2,3,4,7) has the value 0.4, we will still give (2,3,4,7) a priority if such example is seen because there is generally a smaller chance of longer combinations occurring together mid-sentence unless it's actually a single word. The constant 0.2 is added so that monosyllabic words still get a chance to be considered. Through experimentation we have found that 0.2 is the

optimal trade-off point; if this constant is too high, each of all syllables will be considered as a monosyllabic word by itself because they don't have to occur together with some other syllable to be counted. Finally, we divide 1 by this value to transform the occurrences into a cost rather than a reward.

This function will then be used as a cost function for the Uniform Cost Search operation, which will be explained below. This UCS will take the state ((remaining segments), # of iterations). The possible actions will look like the following:

$$state((1,3,2,5,2),1) = actions \begin{cases} 1 & \rightarrow \text{state'}((3,\ 2,\ 5,\ 2),\ 2) \\ (1,3) & \rightarrow \text{state'}((2,\ 5,\ 2),\ 2) \\ (1,3,2) & \rightarrow \text{state'}((5,\ 2),\ 2) \\ (1,3,2,5) & \rightarrow \text{state'}((2),\ 2) \end{cases}$$

Where for each possible action, we're referencing the occurrences dictionary for the cost of that action. However, to keep the UCS computationally efficient, we will "prune the tree" by keeping only the top 3 actions with the lowest amount of cost. This constant can change depending on any context of the data and the syllable number assumptions we're making.

The justification of using a UCS is to go one step forward from the occurrence and to add an extra confidence to the lexical data we are generating by looking at what is a likely *combination* of these words, given the audio stream. For example, given the array of syllable clusters as seen before:

{ 2, 5, 2, 5, 8, 1, 2, 5, 2, 6, 2, 4, … }

If we take a *greedy* approach and simply declare that (2, 5, 2) is a word because of a high chance of it occurring elsewhere in the audio stream, then we would not be considering whether there is a high chance of a word starting with (5, 8 …) to occur in other places as well. If we use Uniform Cost Search, however, the algorithm will be able to find the most optimal combination of syllable assignments to words with the lowest combination of costs.

Then, as a final step, we collect the segments that has been generated by the UCS to build one final lexicon, each syllable combination listed in the order of number of its appearance.

Word reconstruction and Performance Evaluation:

Given the final word list, we print the top 10 elements, which are syllable combinations that are most likely to form a single word. Then for each syllable category within a "word", we will pull a random syllable audio segment that belongs to that cluster, and concatenate those audio segments together to form a single sample "word" utterance. For each word, candidate 5 of example utterances will be provided.

However, the process of evaluation poses a challenge: while for text-based analysis it is easy to test if the output is correct using some objective comparison method on a machine, for

audio outputs it is difficult if our output matches the intended “word”. Therefore, although subjective, human judgement is unavoidable during the evaluation process.

Before we state the performance rates of each algorithm and data sets, we would like to state the evaluation metric. For each word candidate, if more than half of the example word utterance correctly sounds like a i) a single word and ii) word that has been used in this data, than it would be considered valid. For each audio data, we will be considering how many words out of a given lexicon of this audio have been uniquely identified to determine the algorithm’s success rate.

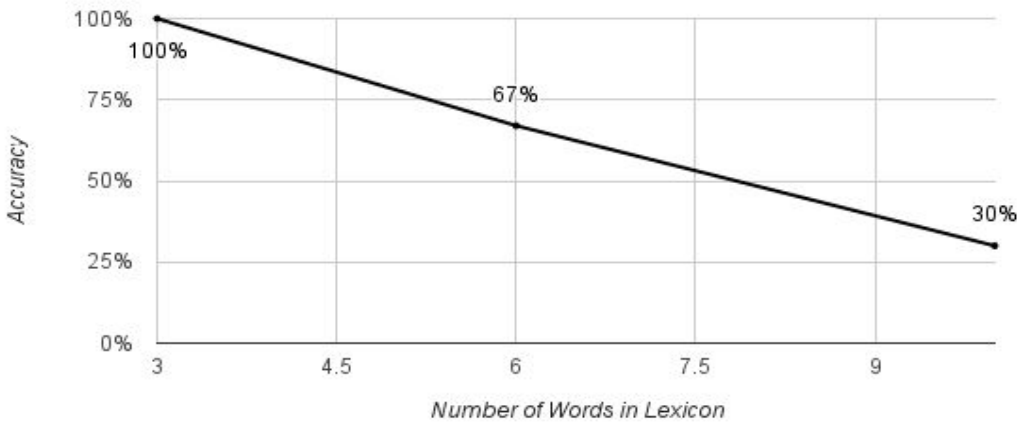
Result:

Algorithm	Accuracy (out of 6)
Breakpoint method	33%
UCS method	67%

The comparison was done on two reference audio data, a 2-minute self-recorded file with 6 English vocabularies. The uniform-cost search (UCS) method provided a much better accuracy than the original breakpoint detection algorithm, so the analysis will be performed on the results returned by the UCS algorithm.

We will be evaluating our upgraded algorithm based on self-recorded data, as it is the easiest way to know the full lexicon of an audio stream beforehand. It was evaluated upon the following metrics:

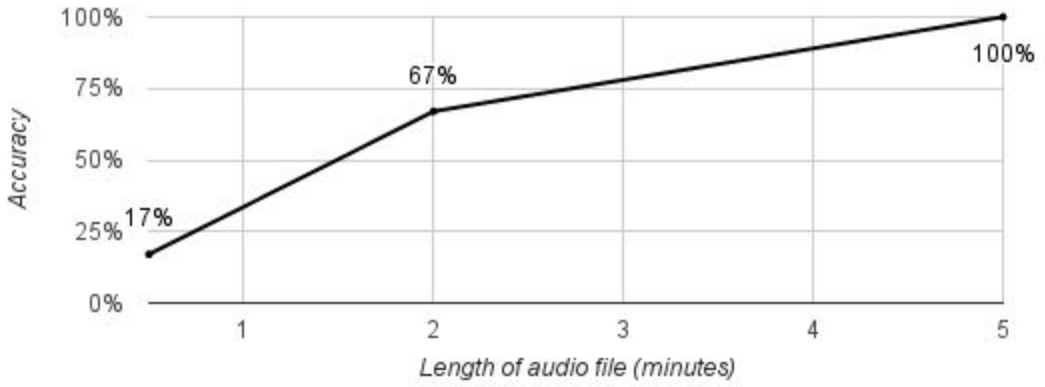
- Number of vocabularies in a lexicon (size of lexicon)



The analysis was done on 3 different 2-minute audio files, each composed of a different

number of words. Increasing the size of the lexicon of the input audio decreased the accuracy with which the algorithm was able to determine the possible words. The words used in each of the audio files were {*dagger, table, guita*r}, {*dagger, table, guitar, panda, sandal, tattoo*}, and {*dagger, table, guitar, panda, sandal, tattoo, beaker, daytime, backpack, laser*}, respectively.

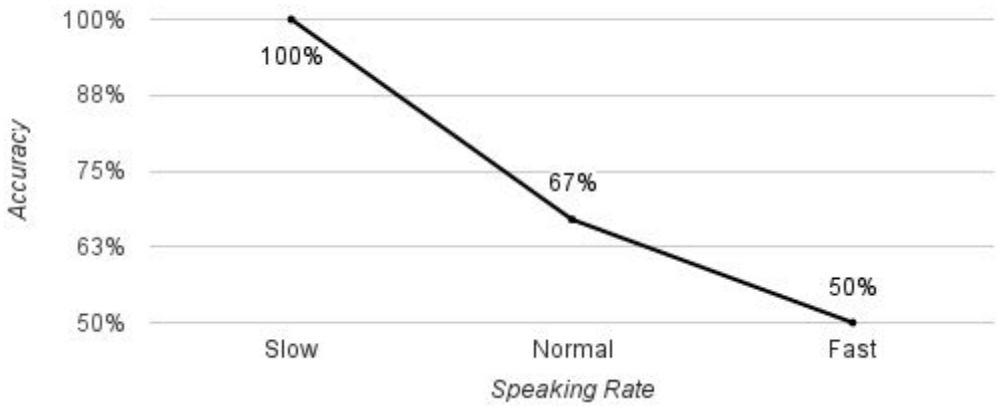
- Length of input audio data



Analysis done on 3 different audio files of varying lengths revealed that the algorithm was more accurate given a longer audio file. Each of these files were composed of the same 6 words, {*dagger, table, guitar, panda, sandal, tattoo*}. The algorithm correctly identified all 6 words when when the audio input was 5 minutes long, but only identified 1 out of the 6 words when the audio input was 30 seconds long.

The above two analyses show why it is incredibly difficult to yield accurate lexical analysis on an arbitrary real-life audio file such as podcasts or audiobooks, even if we assume perfect syllable segmentation, as they contain a high volume vocabulary within a short segment of 6 to 10 minutes without having enough permutations of these words. Considering that infants have to go through at least 3 years to fully achieve lexical analysis, this is not surprising -- if we need accurate results on a real-life audio stream we would need an extremely huge amount of data.

- Speed and fluency of the language speaker



This analysis performed on three different audio files, each 2 minutes long and composed of the same 6 words, **{***dagger, table, guitar, panda, sandal, tattoo***}**, revealed that the algorithm was more accurate when the given audio file was comprised of words spoken at a slower pace. The more the spoken speed increased, the further the accuracy of determining the distinct words from this speech decreased.

- Cross-linguistic analysis

Language	Accuracy (out of 6)	Lexicon
English	67%	{*table, tattoo, dagger, beaker, panda, guitar*}
Korean	67%	{ *엄마, 아빠, 당근, 감자, 개, 사람* }
Hindi	67%	{*kitni, accha, sabun, dosti, khaana, kitab*}

The analysis were performed on 3 different 2-minute audio files, each with words from distinct languages. Their accuracy was surprisingly consistent, as on all of them, our algorithm correctly identified 4 out of 6 vocabularies in each of the files' lexicon. This proves that our initial language-agnostic goal has been met, and that the algorithm performs with consistency regardless of the language and the phonetic structure of the audio file it is being operated on.

Future Challenges:

Syllable detection: 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.

The current onset detection function yields a very low accuracy clustering result for the non self-recorded data sets. However, we must note that syllable -- or phoneme -- detection is a non-trivial problem, and there still is no perfect algorithm to parse an audio stream perfectly into phonological components. This task is made easier by having access to an audio corpus, a.e. Audio files of each phoneme, and manually comparing the corpus data with windows of the audio stream, which is actually a method actively used; however, this algorithm goes against our whole premise in trying to construct a lexicon without any pre-training data or language-specific corpus.

Perhaps, a completely different approach is necessary. Stanford linguist Eve Clark mentions in her book “First Language Acquisition” that there are two competing theories regarding infant language acquisition: the first argues that infants perceive the phonemes of a language first and then build their upper-level lexicon with it, which is the method that we have taken. However, an opposing theory states that infants notice a higher-level pattern in spoken word first before being able to perceive the phonological components behind it. This second approach could hold the key to a more robust model of language acquisition.

Related Work:

- *Saffran, J. R., R. N. Aslin, and E. L. Newport. "Statistical Learning by 8-Month-Old Infants.*

In 1996, Saffran, Aslin, and Newport conducted a study with 8-month old infants using an artificial grammar and found that they could detect word boundaries based only on transitional probabilities. The study sample audio was spoken in a monotone, constant volume. But in real spoken language, prosodic and phonotactic information also helps distinguish between words.

- *Lendvai, Piroska. "Learning to Identify Fragmented Words in Spoken Discourse." N.p., n.d.*

Piroska Lendvai used machine learning algorithms to detect fragmented, disfluent words (such as self-corrections, hesitations, etc.) in spoken Dutch. Algorithms included a memory-based classification system and a rule induction classifier. The machine learning optimizations used for spoken language will be relevant to our project.

- *Clark, Eve V. First language acquisition. Cambridge: Cambridge U Press, 2009.*

As the biggest expert in the field of first language acquisition, Stanford University Linguist Eve Clark discusses the process of infant language learning and suggests that perhaps they recognize the repeated patterns in a language first before being able to perceive the underlying phonology of it.