An introduction to ExemPy

A library for implementing the Generalized Context Model for speech perception

ExemPy is a library for simulating speech perception, and this document will help you understand it. To use spoken language, we have to sort the complex, multimodal, continuous signals of speech into meaningful linguistic categories. As with most complicated topics, researchers disagree about the details of how this happens (Diehl et al. 2004). For example, should we frame the question around sounds, or the movements that produce them during speech? Should we consider perceiving speech to be different from hearing and interpreting non-speech, or is it special to language?

For the rest of this paper, we'll center our discussion in the world of *exemplar* or *episodic* models. You may have heard the term "exemplar theory" to describe a set of distinct (and often mutually exclusive) models of language and other forms of categorization. While it's often useful to refer to what these models have in common, this term can obscure the differences between them, (as categorization usually does!).

ExemPy currently implements one particular model, the Generalized Context Model, or GCM, which I'll describe more in section 2. For now, there are 3 things we need to know about most episodic models of perception:

1. We store memories of past experiences with language
2. Examples are linked to categories
3. Stimuli are categorized based on comparison to examples

To get the most out of this introduction, consider following along with the notebooks linked at each section.

1. What's the point of simulating speech perception?

A simulation of speech perception is not a substitute for behavioral evidence, but it does generate and test hypotheses about behavior.

There are many cases where a perception result would be useful, but isn't possible. For example, when designing experiments, it can be helpful to know things like how similar two sounds are for listeners. If that research doesn't yet exist, a simulation can provide some basis for the design. A sleeping language has no living first-language users, so a well-motivated simulation is the closest we can get to a behavioral result.

With simulations, we also have precise control over each parameter that's been hypothesized. This lets us visualize what behavior would look like if different accounts were true. We can compare these predictions to existing evidence or use it to motivate new investigations.

For example, I'm interested in the data used to argue for "difference in encoding" accounts (e.g., Sumner et al. 2014). These models stipulate fixed differences between exemplars solidified at the time of perception. Could these results also be described using resting activation, *N*, as a flexible parameter that can change dynamically and accumulate?

Implementing a model forces us to make choices that we may be theoretically agnostic to. For example, the code may require two processes to happen sequentially, and so one of those steps has to happen first. A theory may tell us that some value increases, but we have to decide whether to add or multiply. Every parameter requires some value, and it isn't always obvious what that should be.

We accept these limitations because we believe there's value in what they make possible. Every individual decision should be considered with skepticism. As a whole, they work together to give us an impression that's "close enough." That is, useful, if imprecise.

2. What is the GCM?

Recall that "exemplar theory" is a *class* of models, which show a lot of variation. Just like any category, grouping these models together blurs a lot of the important differences and predictions. This is one reason I've chosen to look at a particular, specific model. The Generalized Context Model, or GCM, is an extension of Medin and Schaffer's (1978) context theory of classification (Nosofsky 1986). I use it here to understand speech perception, but this model isn't specific to language. In fact, it was developed to describe visual perception.

The GCM can be summarized using the following equations. I've taken these particular formalizations from Johnson (2006), with only slight notational deviations. What we ultimately calculate is the probability that a language user would categorize the stimulus *i* as a member of category by comparing it to examples, "little-j", of category "big-J". The variable little-j is a stand-in for each stored example we'll be comparing the stimulus to.

There are multiple ways we categorize each exemplar. For example, an object can have both a shape and a color. For each of those category types, big-K, the object has a category label, little-k. So, a green triangle (little-j), has a category label (big-J, little-k) *green* for category type (big-K) *color*, and a category label *triangle* for category type *shape*.

To arrive at the probability, we need to calculate a series of values for each exemplar with respect to the stimulus: distance, similarity, and activation. These values are then totalled, separated by category labels. This gives us the evidence that the stimulus is a member of each category.

(1) $$d_{ij} = \sqrt{\sum_{f} w_f (x_{if} - x_{jf})^2}$$

The distance between two exemplars is calculated based on:

- Values *x* for each feature *f*[1]
- Attention weights *w* for each feature

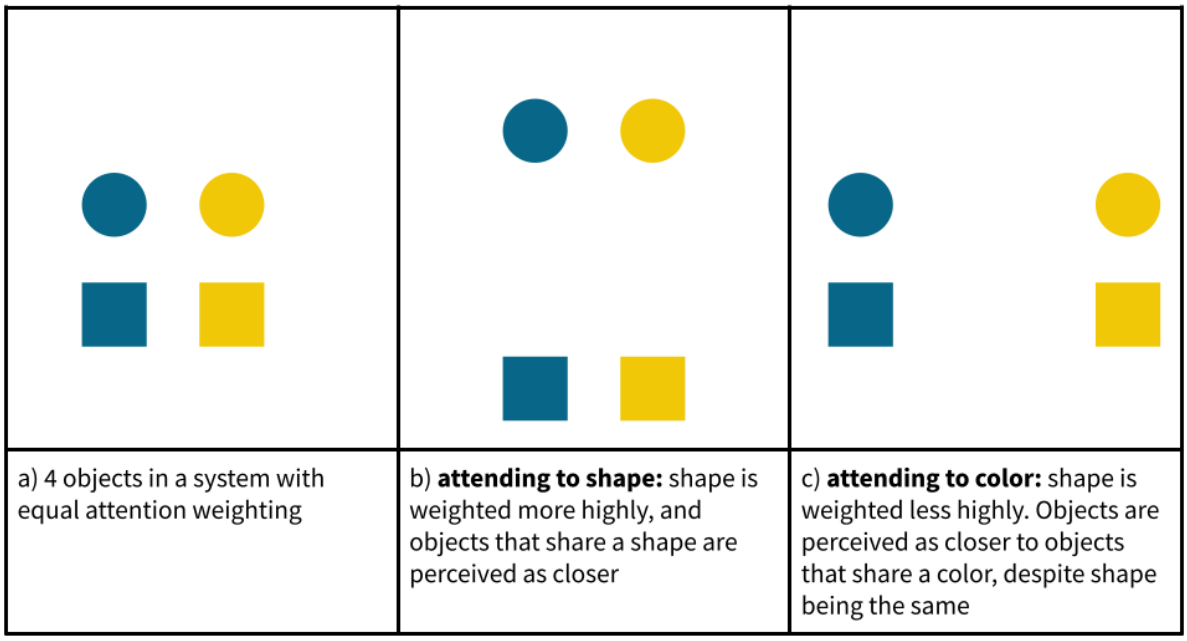
So the term *x_if*[2] refers to the input's value for a given feature, and *x_jf* is the exemplar's value for the same feature. We do this for every feature, and sum those differences up. Squaring the value and then taking the square root gives us the absolute value of the difference. A negative multiplied by itself (a negative) will always be positive. Taking the square root undoes the increase in value. In this way, it doesn't matter whether the value for little-j or i is higher.

Distance is scaled according to how much "attention" is being paid to each feature. These weights are often normalized so that they sum to 1. This constrains the math and makes the relative values easier to compare.

The weights are said to scale the psychological distance between objects based on which features they have in common. It may be easiest to understand this visually. In Figure 1, there are two dimensions of variation: shape and color. Imagine you've been asked to sort these objects into groups, or categories. The answer will change depending on weighting. The more weight is given to a feature, the more similar objects with small differences in that feature will seem.

[1] Elsewhere, you may see this reviewed to as dimensions, *m*

[2] In the interest of legibility, I'll be using underscores as notation for subscripts in the text



a) 4 objects in a system with equal attention weighting	b) **attending to shape:** shape is weighted more highly, and objects that share a shape are perceived as closer	c) **attending to color:** shape is weighted less highly. Objects are perceived as closer to objects that share a color, despite shape being the same

Figure 1: How attention weighting affects distance between objects in a system.

$$(2) \qquad S_{ij} = e^{-cd_{ij}}$$

The distance value calculated in (1) is used to calculate similarity *S*. There are three components of this equation to pay attention to:

- Exemplar sensitivity *c*
- The exponential *e*
- The negative value of the exponent (*-cd_ij*)

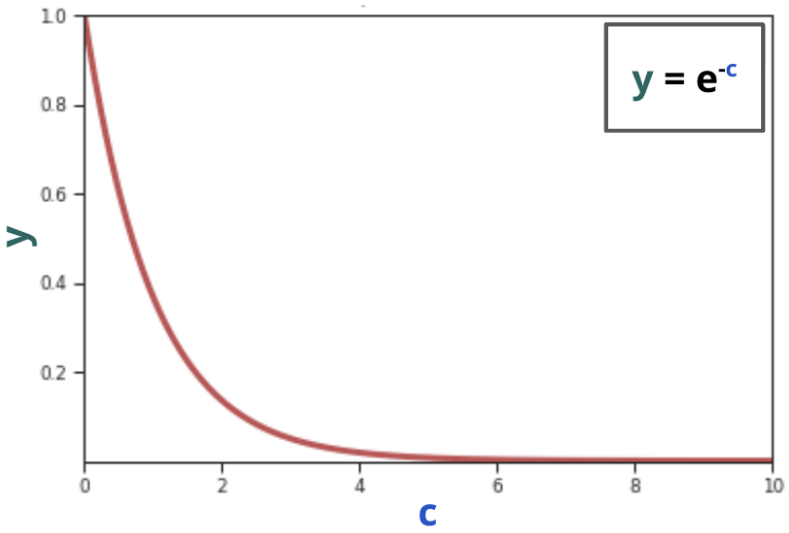


Figure 2: A basic exponential curve, $y = e^{-c}$. As c increases, y decreases. The difference between points also decays as c increases.

One parameter of the GCM is the sensitivity of exemplars, which we'll call *c*. Imagine that each exemplar has a field around it, and the stimulus needs to fit into that radius for the exemplar to become activated.

Taking the exponential of the distance has the effect of distributing distances normally. We can look to Figure 2 to understand the negative value of the exponent: a larger *c* value produces a lower similarity. That is, when *c* is high, a stimulus must be more similar to a given exemplar in order for that exemplar to be meaningfully activated.

$$(3) \qquad a_{ij} = N_j S_{ij}$$

Similarity is used to calculate the activation of each exemplar, multiplied by:

- Base activation *N*

Each exemplar has a starting activation value, *N_j*. This value comes from a listener's expectations about what the stimulus will be. We think this based on things like frequency of the category label, and the likelihood of it to occur in a specific context (Johnson 2006). For example, listeners have a tendency, known as the Ganong effect, to perceive things that are real words. *N* is where that sort of information can contribute to activation.

$$(4) \quad E_{J,i} = \Sigma a_{ij},\ j \in C_J$$

To get the evidence that the input is a member of some category label big-J, we add up the activation of every exemplar little-j that belongs to that label. We do this for every category label within each category type.

$$(5)\ P(R_J|S_i) = \frac{E_J}{\Sigma a_{jk}}$$

The equation in (5) is specialized from a more general axiom known as Luce's choice rule. Remember that ultimately, the category label chosen is the one that was most likely based on the evidence. Luce's choice rule normalizes the evidence relative to the total amount of activation in the system. Let's think through this a little more:

A commonly cited property of exemplar theories is that more frequent categories have a bigger effect on perception. Frequency increases the chances to add activation, producing more evidence for that label. The increase in numerator is scaled by the increase to the denominator. Frequency doesn't overwhelm the effect of similarity. Effectively, a more frequent label needs to be less similar to the input than labels with fewer examples. However, distance and similarity are still the basis of categorization.

2.1 Symbol and term cheat sheet

Exemplar	A stored memory of a past speech perception event
Exemplar cloud	The set of stored memories
i	The stimulus or novel input being categorized
little-j	A particular stored exemplar. Little *j* stands in for every exemplar which the stimulus will be compared to
big-J	The category our stand-in exemplar *J* belongs to
category type (*K*)	Examples of category types are “vowel” or “speaker.” Each category type has multiple labels that could be assigned
category label (k)	The possible categories a stimulus could be labeled with, such as “/u/” or “my grandma.” *J* is an example of a category label
d_ij	The distance between a stimulus *i* and exemplar *j*
w	Attention weights, which scale psychological distance. Different dimensions carry different weight in calculating distance.
x_im	The value of some dimension *m* for stimulus *i*
x_jm	The value of some dimension *m* for exemplar *j*
f	Some feature or dimension things can vary along, such as formant frequencies
S_ij	The similarity between stimulus *i* and exemplar little-*j*
e	The exponential constant
a_ij	The amount of activation that exemplar little- *j* contributes, with respect to the stimulus *i*

a_jk	The activation of each exemplar little-j, divided up by category label
N_j	The resting activation of exemplar little-*j*. An exemplar with a higher *N* value contributes more activation
E_J, i	The evidence that stimulus *i* is a member of a given category label, big-J. Higher evidence for a category label means it's more likely that the stimulus *i* will be given that label

3. What can ExemPy do, and how did I build it?

For detailed documentation of the code, see:
https://github.com/emilyremirez/ExemPy/tree/main/ExemPy

To follow along, see:
https://github.com/emilyremirez/ExemPy/blob/main/ExemPy-Basics.ipynb

Types of simulations

So, if our goal is to compare the results of the model with observable behavior, we begin with reproducing experimental conditions. There are currently two types of tasks which are simulated in ExemPy:

1. Identification tasks, in which the "perceiver" chooses the most probable label, from among all those represented in the exemplar cloud
2. Forced choice tasks, in which the "perceiver" chooses which of two options is more likely

Both of these tasks are simulated using the wrapper function **multicat(),** which categorizes multiple stimuli. As a wrapper function, it calls a series of functions corresponding to smaller steps. Equations (1-3) are implemented as part of the

activation() function. **probs()** implements Luce's choice rule, equation (5), and **choose()** selects the category label with the highest probability as the "percept."

During a forced choice task, an experiment participant is, well, forced to choose between two options. Often, these choices are two clear endpoints of an interpolated continuum. But what does it mean for a perceiver to know that they'll have to pick between two choices? At this time, I identify three, non mutually exclusive vectors for modeling the "side effects" of a forced choice task.

1. During activation()
    - Base activation N: The expectation within the experiment is that the percept will be one of two options. Activation may be raised for the alternatives and quashed for would-be competitors in a straightforward example of priming.
    - Attention weights w: Differences between categories may be emphasized
2. During choose()
    - Rather than choosing the label with the highest probability overall, the "perceiver" chooses the alternative with the higher probability of the two

In the current implementation, probabilities are calculated as usual, but the choose() function chooses between two alternatives.

Setting attention weights

https://github.com/emilyremirez/ExemPy/blob/main/ExemPy-Parameters.ipynb

So, we have hypotheses about the role attention weights serve theoretically–like we saw in Figure 1–but now we need to set actual values. This is one of the cases described in Section 1, where it's unclear what the exact settings of each "knob" should be. To choose a set of values, I used **scipy.optimize.minimize().** I wrote a function that calculates the total error across some category; this is the value we want to make as small as possible. minimize() tries out different combinations of values looking for highest accuracy.

There are numerous possible solutions to this problem, because there are so many variables. (This is sometimes called “cupping” after the multiple low points or “basins” within the solution space.) To address this, I run minimize() multiple times. The function requires you to specify an initial set of parameters to use as a starting point. I generate the initial guess randomly each cycle, trying to get different sets of results. I constrain the possible solutions by setting the first parameter to 1 and limiting the initial guess values to be between 0 and 3.

Once the process is finished, there is another judgment call to be made: Which set of results should we use as the settings? While I generally look for the lowest error value, I also consider other factors. For example, if one parameter is three times the value of the other, I may not choose that result, even if the error is the lowest. Likewise, I might pass if the highest weight is assigned to a cue that I know from experimental work is not very informative to listeners.

Overall, the accuracy for vowel categorization on the Peterson and Barney data has been reliably over 88%. I take this to mean that any set of attention weights generated from the process is as good as anything else. Choosing different parameters will lead to slight variation in results which replicate overall trends.

The attention weights generated by this process are floating point numbers, or numbers with decimal points, rounded to 3 decimal places. During the activation() function, these values are normalized so that they sum to 1. To do this, each number is divided by the sum of all of the numbers.

The exemplar cloud

The **“exemplar cloud,”** or set of stored memories is represented by a Pandas dataframe, which can be created from a spreadsheet or table. Each row represents an exemplar.

There are two types of columns: categories and dimensions. Figure 3 shows a sample of the Peterson and Barney dataframe we’ll be working with throughout this document. In this example, the category types are speaker type, speaker

gender, speaker id, and vowel. The dimensions are fundamental frequency and the first three formant frequencies. You'll notice that Figure 3 also includes the variable repetition. This information helps us uniquely identify each exemplar using a combination of columns, but doesn't really into fit into the category/dimension binary that ExemPy, in its current implementation, assumes.

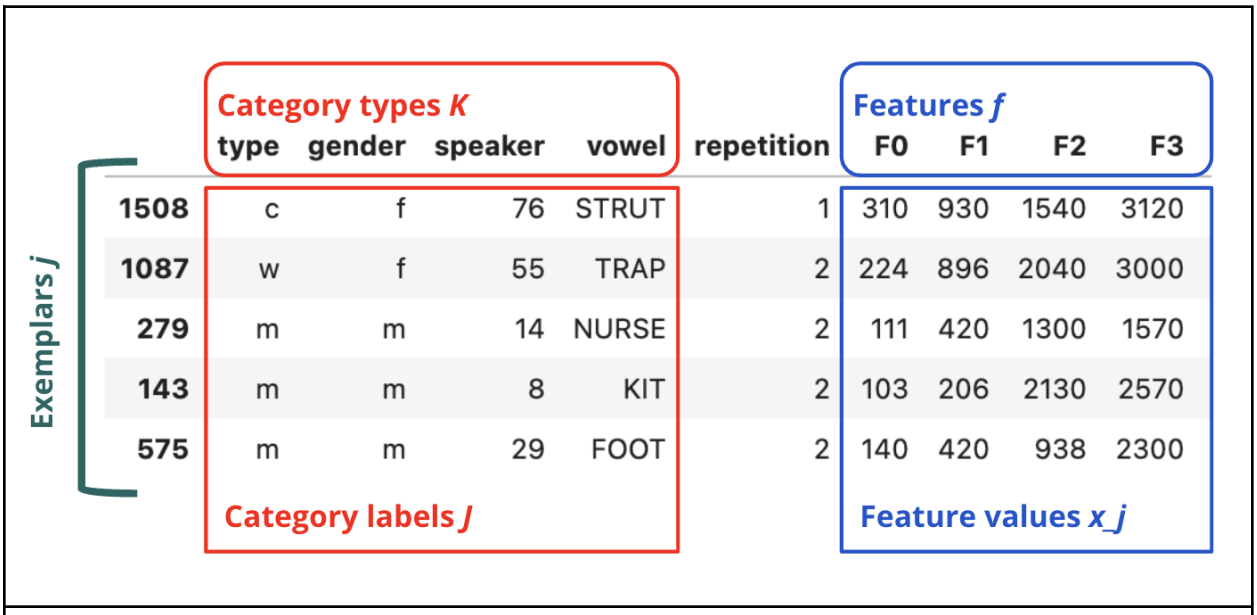


Figure 3: A sample of an exemplar cloud dataframe. Data from Peterson & Barney 1952, via Barreda 2015.

The **stimulus**, *i*, or set of stimuli, should also be a dataframe with at least one row. In the exemplar cloud, every cell must be filled. Each exemplar has a category label for each category type, and a value for each feature. For the stimuli we're categorizing, only the dimension values are necessary. It's crucial that the columns are named identically, and that these same names are used in the dictionary specifying attention weights for each dimension. ExemPy needs to be able to match up this information in order to calculate activation. In most of our examples, we draw the stimuli from the same dataframe as the exemplar cloud. Because of this, we know that the columns will match up.

4. How do ExemPy results compare to behavior?

For each of these demonstrations, I used a c value of 25.

Data

The data in these demonstrations are from Peterson & Barney 1952. One reason I chose this dataset is that other phoneticians will be very familiar with it. It's been used in other simulations, and using an established set of exemplars lets us compare results more directly. In this dataset, 76 speakers repeated 10 vowels twice each. I represent these vowels using lexical set notation; each word stands for the vowel it contains. The speakers are divided into 3 "types": man (33 speakers), woman (28), and child (15). It's important to recognize that the limited understanding of gender this entails oppresses and harms real people.

Identification task

https://github.com/emilyremirez/ExemPy/blob/main/ExemPy-Basics.ipynb
https://github.com/emilyremirez/ExemPy/blob/main/ExemPy-correlations.ipynb

To simulate an identification task, I categorized the entire Peterson and Barney (1952) dataset. Each row in the data frame is compared to every other row, except for itself. You can use the exclude_self argument to turn this off, or the also_exclude argument of activation() or multicat() to specify other exemplars to exclude. That is, you could simulate perceiving a novel talker by not comparing their stimuli to exemplars from the same speaker.

Doing this allows us to compare our results to human behavior. A model is judged not just on how it gets things right but how it gets things *wrong*. That is, when human listeners responded to these stimuli, they sometimes mistook one vowel for another. Informative patterns in these mistakes are captured in confusion matrices. These tables show how often a stimulus was given a particular response. If perceivers regularly confuse one sound for another, that tells us those sounds are perceptually similar.

ExemPy's **confusion()** function will generate a dataframe like the one in Figure 4 below from a dataframe of choices. The columns here represent the stimulus, while the rows represent the response. The diagonal captures accurate identification, and has the highest values. That is, in all cases, the "listener" was more likely to be right than wrong.

vowelChoice	DRESS	FLEECE	FOOT	GOOSE	KIT	NURSE	PALM	STRUT	THOUGHT	TRAP
DRESS	0.84	0.00	0.00	0.00	0.12	0.01	0.00	0.00	0.00	0.03
FLEECE	0.00	0.97	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00
FOOT	0.00	0.00	0.82	0.13	0.00	0.00	0.00	0.03	0.02	0.00
GOOSE	0.00	0.00	0.14	0.82	0.00	0.00	0.00	0.00	0.04	0.00
KIT	0.07	0.05	0.00	0.00	0.88	0.00	0.00	0.00	0.00	0.00
NURSE	0.06	0.00	0.00	0.00	0.02	0.89	0.00	0.00	0.00	0.03
PALM	0.00	0.00	0.00	0.00	0.00	0.00	0.85	0.09	0.06	0.00
STRUT	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.90	0.01	0.00
THOUGHT	0.00	0.00	0.01	0.03	0.00	0.00	0.09	0.02	0.85	0.00
TRAP	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90

a) GCM

	DRESS	FLEECE	FOOT	GOOSE	KIT	NURSE	PALM	STRUT	THOUGHT	TRAP
DRESS	0.658	0.013	0.004	0.001	0.237	0.011	0.000	0.003	0.000	0.072
FLEECE	0.006	0.962	0.000	0.000	0.031	0.000	0.000	0.000	0.000	0.000
FOOT	0.001	0.000	0.620	0.284	0.002	0.009	0.001	0.052	0.031	0.000
GOOSE	0.000	0.002	0.090	0.891	0.002	0.002	0.000	0.007	0.007	0.000
KIT	0.067	0.251	0.006	0.000	0.670	0.001	0.000	0.001	0.000	0.003
NURSE	0.040	0.003	0.030	0.007	0.041	0.866	0.000	0.009	0.000	0.003
PALM	0.002	0.000	0.006	0.001	0.000	0.000	0.550	0.136	0.305	0.001
STRUT	0.009	0.000	0.027	0.001	0.001	0.012	0.128	0.747	0.068	0.007
THOUGHT	0.000	0.000	0.130	0.059	0.000	0.000	0.059	0.080	0.672	0.000
TRAP	0.280	0.001	0.003	0.000	0.006	0.019	0.040	0.020	0.000	0.632

b) Peterson and Barney

Figure 4: Confusion matrices from (a) one round of identification task and (b) Peterson and Barney.

The results of the simulation are highly correlated with the experimental results reported by Peterson and Barney. Again, the exact results will vary depending on the attention weighting parameters, but the results of this example are representative of my other simulations.

To compare the tables, I use two measures. The root mean square (RMS) distance is a measure of the difference between two matrices, while Pearson's correlation (*r*) measures their similarity. If the tables are similar, RMS will be low and *r* will be high. In the exemplar cloud, every cell has to be filled. Each exemplar has a category label for each category type, and a value for each dimension. To calculate these results, I first excluded every 0.0 from the table. Because 0 correlates perfectly with 0, skipping this step would artificially inflate the correlations.

To create Table 1, I fit attention weights for the identification task 5 times. For each set of parameters, z0 (fundamental frequency in Bark) was anchored at 1. Note that the attention weights shown in the table have been normalized to sum to 1, rounded to 2 decimal places.

	Attention weight *w* values					**Comparisons**	
Trial	**z0**	**z1**	**z2**	**z3**	**error**	***RMS***	***r***
1	0.10	0.37	0.23	0.29	0.109	0.073	0.964
2	0.17	0.46	0.22	0.15	0.118	0.075	0.964
3	0.27	0.43	0.16	0.15	0.109	0.074	0.966
4	0.14	0.35	0.26	0.25	0.109	0.073	0.965
5	0.18	0.31	0.29	0.21	0.115	0.075	0.966

Table 1: Comparisons of confusion matrices categorized using 5 different sets of

attention weights

Forced choice task

- [Categorical perception/forced choice demo]

To simulate this forced choice task, I first created three sets of “stimuli” using ExemPy’s **continuum()** function. This function creates interpolated continua between two end points. These 7-step continua started with average values for FOOT on one end, moving in even increments towards the value for STRUT on the other. I did this using average values for the three speaker types: w[oman], c[hild], or m[an].

Figure 4 was created using the **cpplot()** function in ExemPy. It shows the results of categorizing these three sets of continua. The X-axis shows the step along the continuum, with 1 being distinctly FOOT, 7 being distinctly STRUT, and the rest being somewhere in between. The Y-axis represent the proportion of people who would have given that response in a listening experiment. When FOOT is The gray dotted line at 0.5 visualizes chance. The point at which the curve intercepts the 0.5 line is considered the “boundary” between those two categories. This graph shows that our “listener” draws the boundary between FOOT and STRUT differently for the man stimuli than for woman and child.

Remember that ultimately the more similar an exemplar is to the stimulus, the more likely it is that the stimulus will be categorized the way as the exemplar. Based on past experience, listeners know that the difference between a man FOOT and STRUT isn’t the same as the difference between FOOT and STRUT at generally higher frequencies, as associated with women and children.

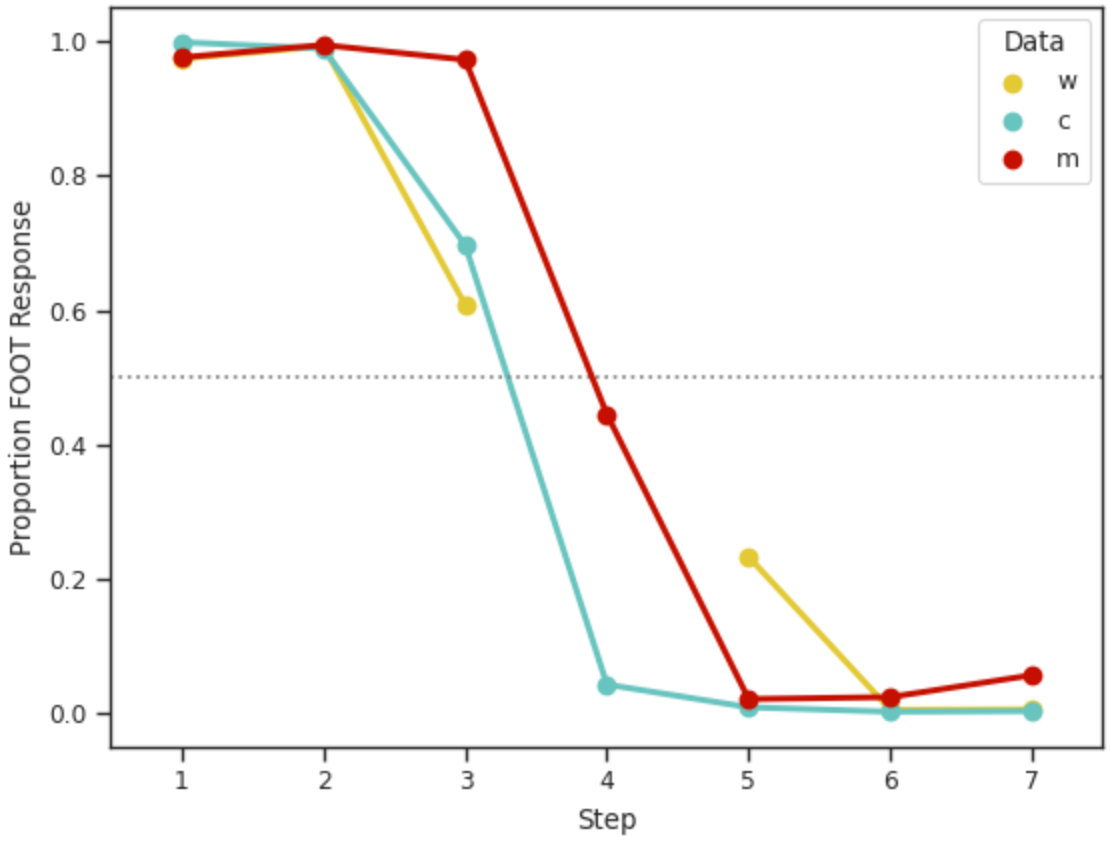


Figure 4: Results of simulated forced choice task

In Figure 4, there is no point for step 4 in the woman stimuli. This is because the most probable response was neither FOOT nor STRUT, but PALM. To plot any point would imply an artificially high probability of one of the two options.

5. That's it!

This work owes an incredible debt to Keith Johnson, Susan Lin, Kevin B. McGowan, Terry Regier, and Ronald Sprouse. I did this work on the unceded land of the Ohlone people, whose relationship with the land continues to this day.

6. Bibliogrpahy

Barreda S (2015). *phonTools: Functions for phonetics in R.*. R package version 0.2-2.1.

Diehl, R. L., Lotto, A. J., & Holt, L. L. (2004). Speech perception. Annual review of psychology, 55(1), 149-179.

Johnson, K. (2006). Resonance in an exemplar-based lexicon: The emergence of social identity and phonology. Journal of phonetics, 34(4), 485-499.

Medin, D. L., & Schaffer, M. M. (1978). Context theory of classification learning. Psychological review, 85(3), 207.

Nosofsky, R. M. (1986). Attention, similarity, and the identification–categorization relationship. Journal of experimental psychology: General, 115(1), 39.

Peterson, G. E., & Barney, H. L. (1952). Control methods used in a study of the vowels. The Journal of the acoustical society of America, 24(2), 175-184.

Sumner, M., Kim, S. K., King, E., & McGowan, K. B. (2014). The socially weighted encoding of spoken words: A dual-route approach to speech perception. Frontiers in psychology, 4, 1015.