The parallel distributed processing approach to semantic cognition: a comparative extension to a complex domain

Sheetal Laad
NYU CDS
sl7054@nyu.edu

Orion Taylor
NYU CDS
ojt212@nyu.edu

Ziv Schwartz
NYU CDS
zs1349@nyu.edu

Yash Deshpande
NYU CDS
yd1282@nyu.edu

Abstract

This paper expands upon our earlier implementation of a neural network model for semantic cognition. Semantic cognition describes the cognitive mechanism by which humans develop and store object-property associations in long-term memory, and our subsequent ability to generalize from these characteristics to "identify" objects in terms of trait similarity with previously-observed objects. We engineer a larger and more complex data-set for training within a baseline parallel distributed processing (PDP) architecture, exploring the nature of semantic cognition in the domain of food. Additionally, we employ an identical architecture with a higher degree of complexity. Lastly, we contrast the modeling of semantic cognition in our domain against that used by McClelland and Rogers (2003).

1 Introduction & Literature Review

This project is an extension of our initial exploration of the model of semantic cognition proposed by McClelland and Rogers (2003). We found mutual interest in this topic, and in whether our cognitive interpretation of more complex domains could be represented using the same model (and/or whether a more complex model could outperform). In the model postulated by McClelland and Rogers (2003), human semantic cognition is approximated using a neural network; the archetypal example uses a simple neural network and a small set of real organisms (generally nouns), relations (generally verbs) and attributes (nouns, adjectives or verbs, depending on activated relations). The architecture of their model is shown in Figure 1.

The computational design of this network includes an encoding process; input items are expressed in a pseudo-binary capacity and passed into a linear "Representation" layer using rectified linear unit activation (ReLU), bundled with relational features (e.g., "Canary CAN") into a single string of 0s and 1s, passed through a single linear hidden layer with ReLU activation, and finally into a linear output layer with Sigmoid activation of corresponding attributes. Theoretically, after training a model in this fashion it should be able to simulate the process of generalization by predicting the activation strengths for objects whose properties have not been directly observed.

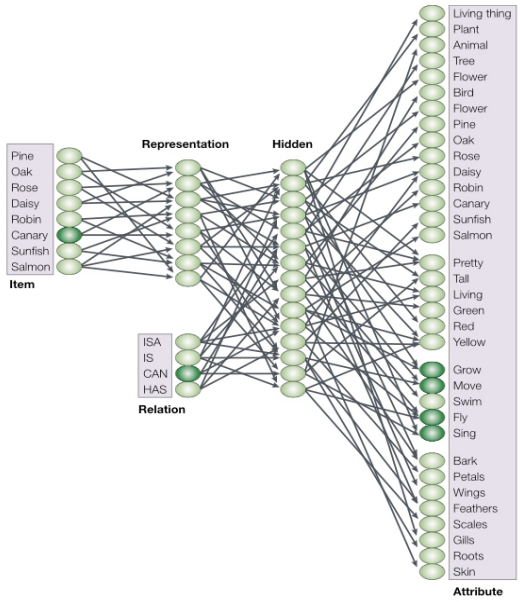


Figure 1: McClelland and Rogers (2003) model architecture

We wanted to find a domain that was at once complex (in terms of data volume, number and division of input features, and attributes) and straightforward (i.e., distinguishable inputs, simple attributes, easily interpreted, etc.). After some research and discussion, we settled on food dishes, and found a suitable data-set from the Kaggle challenge What's Cooking?. The data consists

of JSON-formatted lists of ingredients from unlabeled recipes tagged by cuisine-type, as illustrated in Figure 2 below.


```
{
"id": 24717,
"cuisine": "indian",
"ingredients": [
    "tumeric",
    "vegetable stock",
    "tomatoes",
    "garam masala",
    "naan",
    "red lentils",
    "red chili peppers",
    "onions",
    "spinach",
    "sweet potatoes"
]
},
```


Figure 2: Raw data

Subsequently, we focus on recipes from 20 distinct cuisines, hoping to observe a modeled process of differentiation between them through dendrogram visualization of their similarity (in terms of ingredient(s) co-occurrence).

2 Methodology

2.1 Data

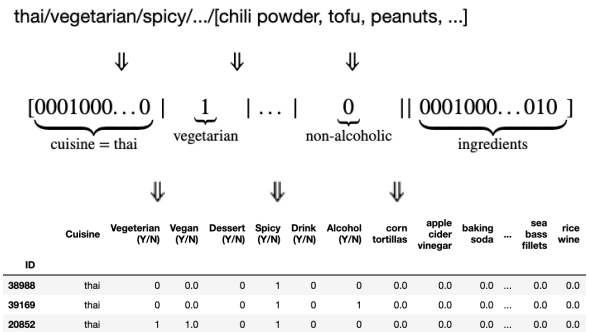
Our initial recipe data is obtained from What's Cooking?, and consists of $39,774$ common recipes belonging to 20 distinct cuisines. These include *Greek, Southern US, Filipino, Indian, Jamaican, Spanish, Italian, Mexican, Chinese, British, Thai, Vietnamese, Cajun-Creole, Brazilian, French, Japanese, Irish, Korean, Moroccan,* and *Russian*.

Each row of the data consists a unique identifier, a cuisine label, and a list of ingredients. We incorporate a randomized selection of 10 recipes from each cuisine. In determining additional labels with which to tag each recipe, we may introduce an element of subjectivity, and therefore err towards conservative application. For example, manufacturing a tag like *healthy* might incorporate our 4 unique definitions of healthiness and therefore cloud any objectivity the model requires to differentiate between cuisines, whereas an indicator for the presence of meat is perfectly straightforward and produced uniformly. Our final set of labels, in addition to *Cuisine*, are *Vegetarian(Y/N), Vegan(Y/N), Spicy(Y/N), Dessert(Y/N), Drink(Y/N),* and *Alcoholic(Y/N)*. These are henceforth referred to as *taste* labels.

On review of the recipes, it becomes apparent that there are certain ingredients, though identical for the purposes of this analysis, are described differently between the various recipes. For example, $\{$*Hot water, cold water, warm water*$\}$, $\{$*low-sodium chicken broth, chicken broth*$\}$ are some sets of *distinct* ingredients observed in the data. Since many of the differences in the ingredients are descriptors of the ingredient(s) themselves, removing the descriptors does not change the value added by the ingredient itself. The data is stripped of the descriptors, resulting in a larger number of common ingredients between cuisines. Following the earlier example, *hot water, cold water,* and *warm water* are all changed to a single ingredient label- *water*. In contrast, certain ingredients are allowed to retain distinction- e.g. *red pepper* and *green pepper* are changed to *bell pepper*, whereas *black pepper* remains distinct.

Without descriptor changes the data consists of 200 recipes, with 779 unique ingredients. Once the descriptors are stripped, the latter count is brought down to 606 unique ingredients. Henceforth, these data sets are referred to as *non-condensed* and *condensed* respectively. This data is formatted as shown in Figure 3 below.

Additionally, in order to determine the affect of sample size on model performance, we split this data into two sets. The first, henceforth referred to as the *small* data set, consists of a randomized selection of 5 recipes from each cuisine. The second, full data set is referred to as the *large* data set.



ID	Cuisine	Vegeterian (Y/N)	Vegan (Y/N)	Dessert (Y/N)	Spicy (Y/N)	Drink (Y/N)	Alcohol (Y/N)	corn tortillas	apple cider vinegar	baking soda	...	sea bass fillets	rice wine
38988	thai	0	0.0	0	1	0	0	0.0	0.0	0.0	...	0.0	0.0
39169	thai	0	0.0	0	1	0	1	0.0	0.0	0.0	...	0.0	0.0
20852	thai	1	1.0	0	1	0	0	0.0	0.0	0.0	...	0.0	0.0

Figure 3: Transformed data

2.1.1 Pre-processing

Once the data is labeled and cleaned, it is binarized for PyTorch readability. This is done by creating a

one-hot encoded list of labels for each recipe. The first 20 digits represent the cuisine label, and the next 6 represent the *taste* labels.

The ingredients are binarized in a similar manner. To draw an analogy with the *Item - Relation - Attribute* architecture (Figure 1) incorporated by McClelland and Rogers (2003), our exploration interprets *cuisine* and *taste* labels as *items*, and *ingredients* as *attributes*. We choose to eliminate the *relation* layer; in the context of this domain, such a split does not seem natural.

Our exploration differs from that of McClelland and Rogers (2003) in two significant ways: (1) the domain of the data, and (2) the elimination of a *distinct* relation layer.

2.2 Modeling

2.2.1 Baseline: feed-forward neural network

The baseline model is similar to that described by McClelland and Rogers (2003), the only change being the removal of a separate relation layer for reasons described in Section 2.1.1.

This consists of four linear layers: one input layer, two hidden layers (called *representation layer* and *hidden layer* respectively) and an output layer. The binarized data described in Section 2.1 above is fed to the network. The first 26 entries in each row are input patterns that the network uses to learn; representations of these *items* are learned using their mappings to the *ingredients*, which are the output patterns or ground truth labels.

ReLU activation is used for the *representation* and *hidden* layers. The output layer, however, makes use of Sigmoid activation. To clarify, the *output pattern* can be thought of as a predicted set of *ingredient* labels.

2.2.2 Extension: an added hidden layer

To add a level of complexity to our analysis, we employed a second network architecture, which consists of a third linear hidden layer. This also makes use of ReLU activation.

2.2.3 Parameter settings

Identical parameter settings are used for both the baseline and extension models, as shown in Table 1 below.

Parameter setting	**Value**
rep_size	26
hidden_size	26
learning_rate	0.1

Table 1: Parameter settings

3 Results

3.1 Optimization results

We train both the above models on two broad categories of data: *condensed* and *non-condensed* (as described in Section 2.1). Additionally, under each category, we train the network using both the *small* and *large* data sets.

Each network is trained for 5000 epochs, with the parameter settings described in Table 1. *Mean squared error (MSE)* is used for optimization. *MSE* as well as the outputs of the *representation* layer are extracted after 500, 2500, and 5000 epochs respectively. The *MSE* values over epochs for both models are tabulated in Tables 2 and 3 below.

Data/Epoch count	**1**	**2500**	**5000**
Non-condensed (small)	0.25	0.019	0.019
Non-condensed (large)	0.25	0.013	0.013
Condensed (small)	0.25	0.025	0.021
Condensed (large)	0.25	0.016	0.015

Table 2: Mean squared error loss - baseline model

Data/Epoch count	**1**	**2500**	**5000**
Non-condensed (small)	0.25	0.019	0.019
Non-condensed (large)	0.249	0.013	0.013
Condensed (small)	0.251	0.026	0.023
Condensed (large)	0.25	0.016	0.015

Table 3: Mean squared error loss - extension model

3.2 Analytic results

Each permutation of *data set type* / *model architecture* was tested. The representations for each item (*cuisine*) were extracted after having trained each model for 500, 2500, and 5000 epochs respectively.

These representations were visualized using the dendrogram function from the Python *SciPy* library. The patterns are compared in order to find the two that are closest to each other by virtue of a Euclidean distance measure. Subsequently, it iterates over the patterns, replacing each closest pair with an average over that pair, until an underlying pattern remains.

In the interest of concision, all dendrograms can be found in Appendices A and B; merely those immediately relevant to our analysis will be part of the main text in coming sections.

4 Analysis and Conclusion

The PDP framework employed by McClelland and Rogers (2003) is based on a model of semantic cognition, and lends itself to the idea that with successive development in the brain, the general property of a particular item comes to be associated with said item more strongly than a specific property.

Human beings learn to differentiate between objects by way of example, storing certain generalized representations of their attributes that aid in categorization, when required. For example, a child that has only seen 2 distinct bird species in her lifetime, both flighted, would make the reasonable assumption that the object category *bird* must have the attribute *flight*. However, an adult, presumably exposed to a larger number of distinct bird species, may not make the same generalization.

The nature and domain of the data used by McClelland and Rogers (2003) lends itself naturally to this framework of learning. They find that early patterns (at, say, 500 epochs) are strikingly undifferentiated, with the first difference to appear being between plants and animals. As learning progresses, this differentiation becomes increasingly prominent (i.e. with more examples and more experience with object attributes). At the conclusion of the learning process, individual items are differentiated while retaining the overall hierarchy.

The domain we explore, however, is significantly different. Whereas that explored by McClelland and Rogers (2003) exhibits universal *item - attribute* relationships, the domain of food is in many ways less objective. Given a set of ingredients and asked to attribute this with a *cuisine* label, an individual's response may (1) be biased toward cuisines they have been exposed to, and (2) not be a single cuisine, as individual ingredients are (for this purpose) atomic and far removed from any generalization of a cuisine.

The dendrograms in Appendices A and B represent patterns of differentiation for each data set type / model combination, similar to those seen in the paper by McClelland and Rogers (2003). We see that the level of differentiation observed by McClelland and Rogers (2003) is not visible here; however, there is still *some* semblance of differentiation. Looking at the graphs for 5000 epochs, i.e. at the conclusion of training, we see that the model performs slightly better when trained on the larger data set. Additionally, differentiation is more clearly seen when the *condensed* data set is used- this is logical, since we expect that the model learn generalizations of cuisine attributes (or ingredients).

The extension of the baseline model, i.e. that with an additional hidden layer, does only marginally better (as seen from the MSE in Table 3). In terms of cuisine attribute differentiation, however, the added layer seems to perform worse (Fig. 5 vs. Fig. 7 and Fig. 9 vs. 11).

We attempted to incorporate a relation layer, with the aforementioned *taste* attributes in place of the *relation* attributes used by McClelland and Rogers (2003). However, this is not a natural *Item - Relation - Attribute* progression, and the network failed to learn differentiated representations. We chose to exclude these results from our report, as they are insignificant.

As an expansion to this project, we would create a visualization of the input-to-output layer activation paths similar to the one created by McClelland and Rogers (2003) (Figure 1). In addition to extracting representations for each item (cuisine), we could experiment with a softmax activation layer to output probabilities for ingredients given specific cuisines. The manufactured features (spicy, vegetarian, etc.) we included as inputs would likely help to refine such probability scores, since they might enable more accurate differentiation among items (e.g., the likely ingredients for a generic *Mexican* item might not be as strong as for a spicy Mexican drink). This notion parallels human information accumulation. Another interesting point would be to compare such predictions with empirical data collected from human participants. This would be an interesting area to analyze further through more experimentation and additional development of the current model.

References

James L. McClelland and Timothy T. Rogers. 2003. The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience*, 4(4):310–322.

What's Cooking? Kaggle.

A Appendix: Plots (non-condensed data)

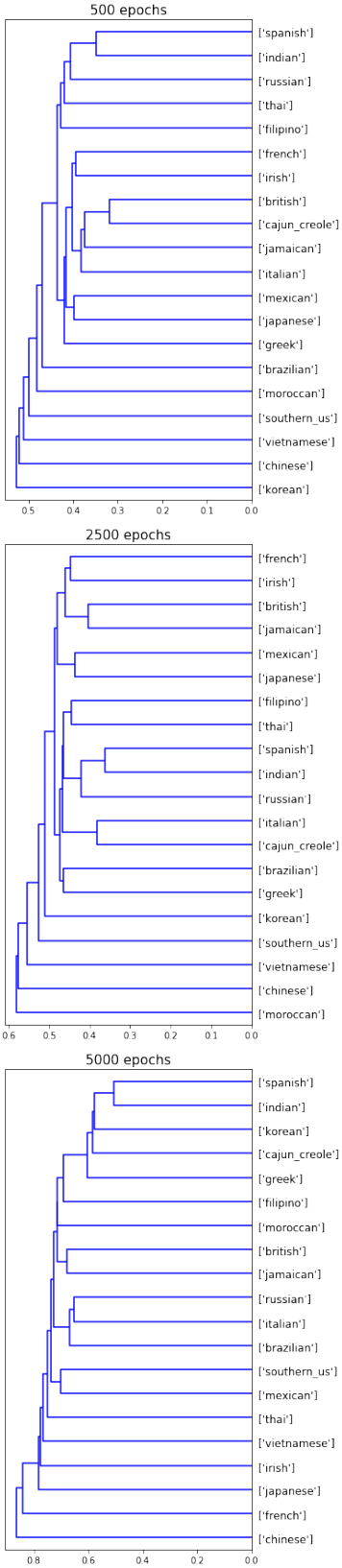


Figure 4: Dendrograms of item representations (baseline model - small data set)

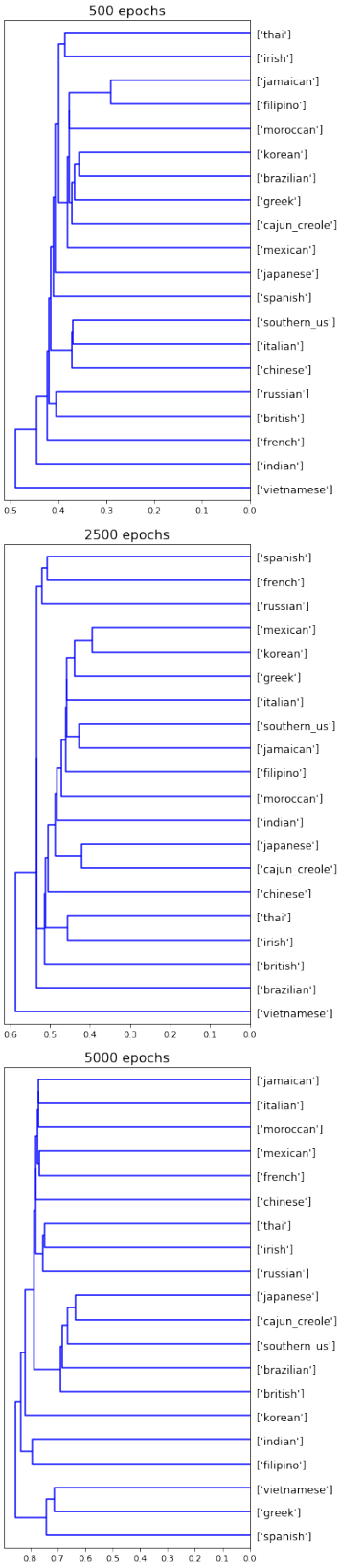
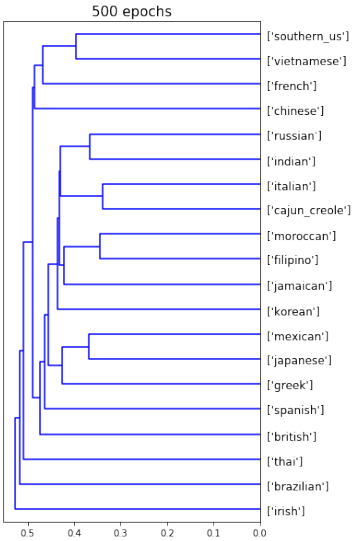
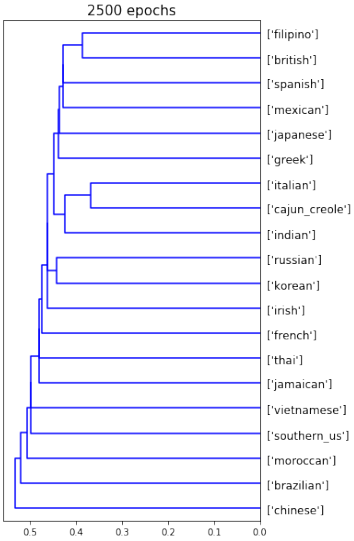


Figure 5: Dendrograms of item representations (baseline model - large data set)





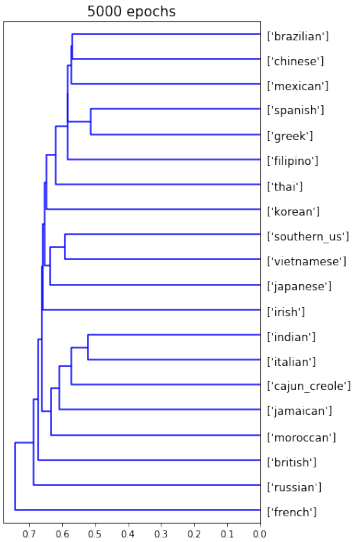
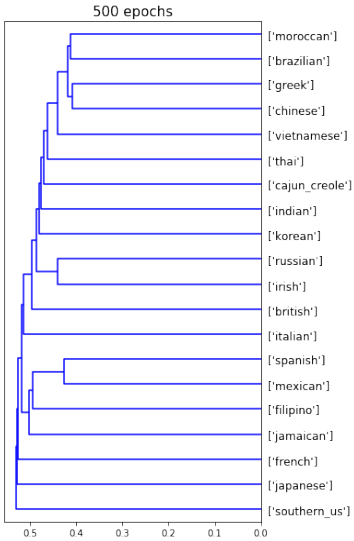
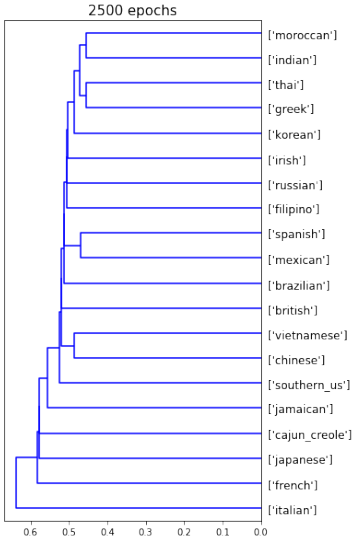


Figure 6: Dendrograms of item representations (extension model - small data set)





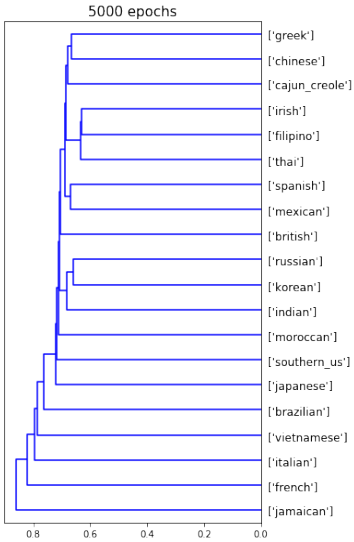


Figure 7: Dendrograms of item representations (extension model - large data set)

B Appendix: Plots (condensed data)

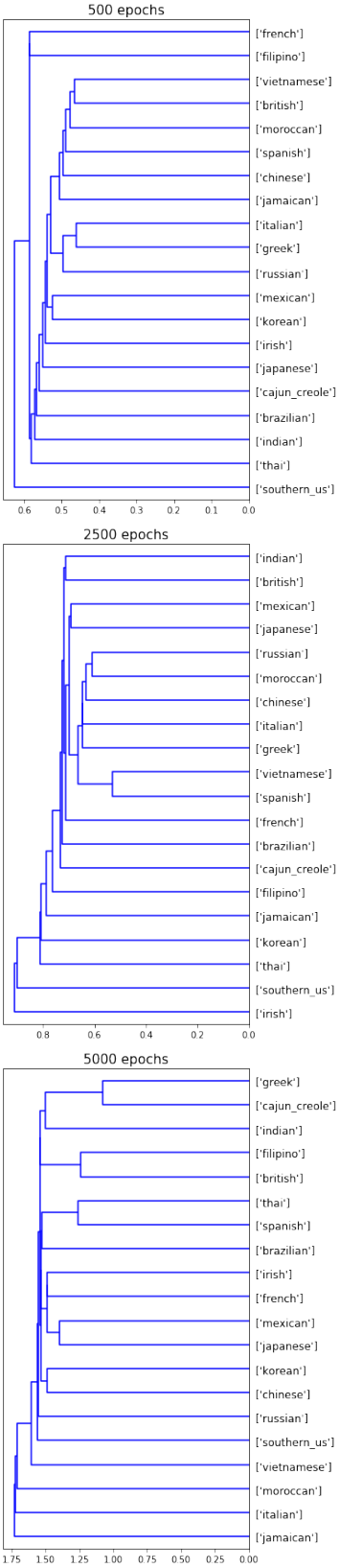


Figure 8: Dendrograms of item representations (baseline model - small data set)

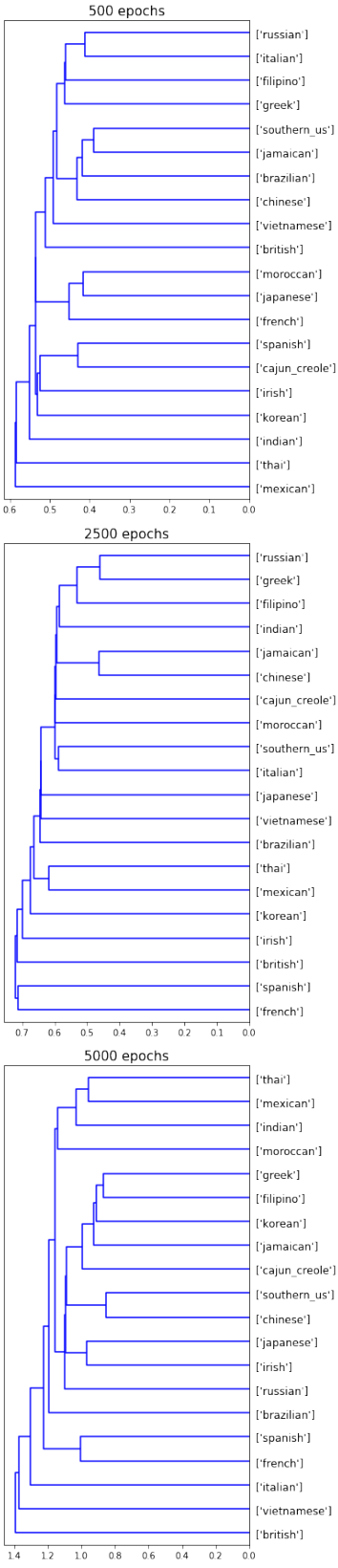


Figure 9: Dendrograms of item representations (baseline model - large data set)

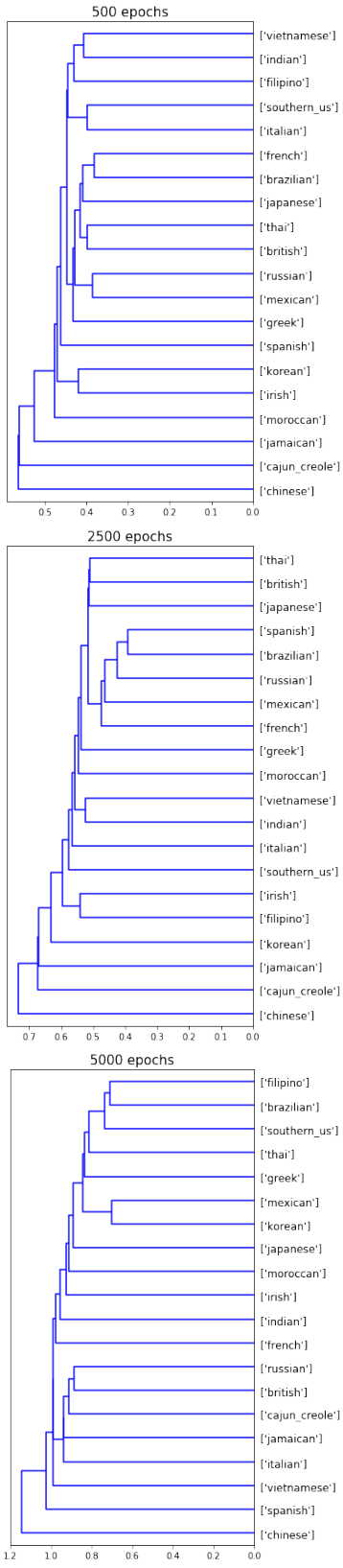


Figure 10: Dendrograms of item representations (extension model - small data set)

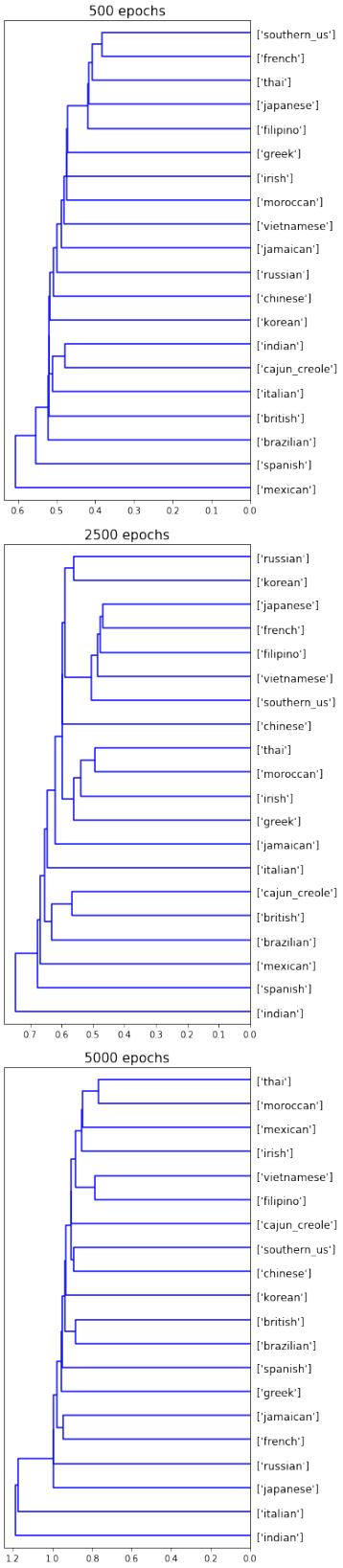


Figure 11: Dendrograms of item representations (extension model - large data set)

C Appendix: Code

The code used for this project can be found in this GitHub repo.