Semantic Cognition and Network Deterioration

DS-GA-1016 Computational Cognitive Modeling
https://github.com/petersim1/CCM_Final_Project

Peter Simone (ps4021)
Maria de la Paz Vives (mdv325)
Qixiu Fu (qf278)

May 13, 2020

1 Introduction

In semantic cognition, people need to know what objects possess what kinds of properties, and to what extent these properties can be generalized to other objects. It is our intuitive understanding of objects and their properties. Past research used a hierarchical model to capture these semantic relationships from specific to general [3]. This model is good for its economy of storage, and can make powerful inference when a new concept is added. However, it does not handle exceptions well, and question arises when deciding which super-ordinate should be included and which properties should be stored within [1]. Instead, McClelland and Rogers took a parallel distributed processing approach to semantic cognition, and investigated how the process of learning and generalization of properties can be affected by brain disorder, like semantic dementia [1]. This parallel distributed model does not store knowledge explicitly in a hierarchy, but implicitly in the weights of the network.

Semantic dementia, a.k.a semantic variant primary progressive aphasia (svPPA), is a progressive neurodegenerative disorder characterized by loss of semantic memory in both verbal and non-verbal domains. Past research on semantic dementia patients found a gradual memory loss in picture naming from specific to general, in which the patients had a more specific vocabularies of duck, swan, chicken at the early stage of disorder, but gradually limited to a more general categories like animal, vehicle at later stage with incorrect naming [1].

In recent years, researchers have considered the gradual learning characteristics of neural networks as capturing the properties of the semantic learning system in the neocortex, and by introducing noise to the neural network, McClelland and Roger provided a basis for understanding the progressive loss of semantic knowledge in semantic dementia [1]. They found the model's internal representations were increasingly degraded, showing a similar pattern of knowledge disintegration to semantic dementia patients.

2 Neural Networks

There are three key principles of neural network models of cognition: 1) it is neurally-inspired computation, 2) intelligence is an emergent phenomenon, and 3) simulation is central for understanding intelligence. A feed-forward neural network generally consists of an input layer, one or more hidden layers, and an output layer. Learning in a neural network is understood as optimizing an objective function in an iterative manner, by updating weights and biases to minimize the loss incurred in the network.

McClelland and Roger built a parallel distributed processing network to construct the process of learning propositions of concepts [1]. Their network was trained to answer queries involving an Item and a Relation, outputting all attributes that were true of the item/relation pair. An example query will be: A pine (Item) has (Relation) roots (Attribute).

In the original paper, they included 8 items (Pine, Oak, Rose, Daisy, Robin, Canary, Sunfish, Salmon), 4 relations (ISA, IS, CAN, HAS), and 36 attributes. Their feed-forward network consists of an input layer (Items and Relations), two groups of hidden layers (representation and hidden layers), and output layer (Attributes). Each layer has an activation function that determines the strength of forward going signals. Learning depends on minimizing the difference between the target and obtained activation of the output layer. Back propagation of error adjusts the weights only - as they chose to make bias "untrainable" - and as weights adapt slowly, it performs a gradual evolution of the patterns of activation and reduction of error [1].

To study the process of Semantic Dementia, noise was added to the representation layer of the neural network, and they found a similar pattern of losing the specifics but preserving the general information like in semantic dementia patients. In this study, we were interested in implementing noise in two ways to a similar neural network model to test whether or not we can replicate a similar finding to McClelland and Roger's paper [1].

3 Methods

3.1 Data Set

We borrowed the same item lists from the McClelland and Roger paper, and introduced new reptile items (turtle and crocodile). This also led to an expansion of attributes in comparison to the original data set (added shell, produce eggs etc.). There was an overlap of attributes among items. For example, turtle(Item) can(Relation) produce eggs(Attribute), and canary(Item) can(Relation) produce eggs(Attribute). There were a total of 10 items, 4 relations, and 43 attributes. All input items were one-hot encoded, making it a multi label classification task. Based on where the attributes fall in the given level of hierarchy, we classified an attribute as "general", "intermediate", or "specific", which will be important in a later section.

3.2 Model Training

While maintaining the same general network architecture, it was adapted slightly for our purposes. Pytorch was used to implement all work related to building the model framework, with custom scripts implemented to train and evaluate the network [2]. There were 10 nodes in the item input layer and 4 nodes in the relation input layer. We upped the representation layer to 10 nodes, and had 15 nodes in the hidden layer. Both representation and hidden layers used ReLu activation functions, and the final output layer used a Sigmoid activation function. In the original paper, the bias in all layers were initialized to -2, and were not learned [3]. We wanted to take a similar approach to this to ensure that all the signal and differentiation observed was originating from the node-to-node learned weights. This is because, when quantifying "differentiation", only the weights are taken into account. Therefore, we initially isolated the learning to only be from within the weights, thinking it would lead to more obvious patterns without the bias correcting factors. However, through trial-and-error of training the model, we determined it was optimal to have no bias in the representation layer - to clearly observe learned hidden representation within and between items - but to actually allow for learnable bias in the hidden and output layers. To handle this, we removed the bias in the representation layer, and then initialized the learnable biases to all 1's for the hidden and output layers. This optimized for clear differentiation over time within the representation layer, but also allowed for high accuracy in the output layer.

The model was trained using BCELoss, which is typically more suitable for multi-label classification than MSE. In BCELoss, each label of the output represents a probability that it belongs to the positive class, and minimizing the BCE loss function essentially serves the same purpose as maximizing the log likelihood. The reasoning for switching to BCELoss was that we were unable to achieve a consistent signal and acceptable accuracy while using MSE. In MSE, despite tuning hyperparameters and trying different network architectures, the data was never truly "learned", meaning there were always missed classification in the output. In BCELoss, the network consistently and successfully identified all appropriate labels in the output.

Lastly, the model was trained using mini-batch gradient descent, to improve training efficiency while approximating pattern-wise updating [1]. Through tuning hyperparameters using stochastic gradient descent (SGD), we noticed that the model was capable of producing high accuracy, but the representation layer was not differentiating as expected. Interestingly, the network was unable to simultaneously acquire the ability to make the correct classifications while also detecting hierarchy in the data. Batch Gradient Descent, on the other hand, was not sensitive enough to expected representation layer differentiation (couldn't detect proper hierarchy) or the ability to call the correct classes. Randomly permuting the training patterns in each epoch, and performing Minibatch Gradient Descent (even of just batch size of 2), was a happy medium in leading to perfect multi-label classification and expected representation layer differentiation and hierarchical structure. A learning rate of 0.05 for 3,500 epochs was optimal and led to significant signal even in the more specific targeted attributes.

3.3 Network Noise / Dropout

To replicate the idea of deterioration of neurons in the brain during a semantic cognitive disease, we simulated deterioration of our network through the use of two different methods. The first one was the addition of random noise, sampled from a Gaussian distribution, to the learned representation layer. This was conducted many times at given "levels" of deterioration, where a "level" of deterioration is denoted as the variance in the Gaussian distribution from which we sample from. For each level of deterioration, a random matrix is generated from a Gaussian distribution, the random weights are added to the final model representation layer weights, the network is updated, and the inputs are run through the newly updated network. After each iteration, the network is reset to its original state. This is performed 50 times at each level of deterioration.

The second method of implementing noise was through node dropout. We could evaluate the dropout effect with a similar approach to what was just mentioned, by updating the learned representation layer then resetting it after each iteration. For each level of deterioration, rather than sampling from a Gaussian distribution to generate randomness, each node of the representation layer has a given probability of dropping out following a Bernoulli distribution. Now, dropping out should not be thought of as altering the architecture of the network. Since our model already has no bias at the representation layer level, dropping out will instead be defined as setting all weights to a given node to zero (which will then make it's output zero as well, given no bias in the representation layer). The different probabilities of nodes dropping out are considered to be the levels of deterioration of the network.

We then evaluated the effect of network deterioration on the expected outputs, broken down into groups representing where in the hierarchical structure their attributes lay, as previously mentioned. "Negative" attributes were also evaluated, so we can observe if signals from previously negative, and unexpected, classifications start to achieve meaningful signals in the network (i.e. will a bird start to be classified as a fish through adding noise).

4 Results

4.1 Differentiation Through Learning

With added items and expanded attributes, compared to the reference paper [1], our training results demonstrated a similar learning pattern to the original paper. We extracted the weights of the representation layer to examine the differentiation of items, and used the learned outputs at each epoch to evaluate the signal in each attribute. In general, differentiation emerges as training progresses. Items belonging to a similar family have a similar weight pattern in the representation layer. In Figure 1, a hierarchical clustering was used to determine how items are related to each other in terms of the learned network representation. Our model first made a differentiation between plants and animals around 1000 epochs. At the end of the training, the model identified five distinct clusters, bird(robin, canary), flower(rose, daisy), tree(pine, oak), fish(sunfish, salmon), and reptile(turtle, crocodile), while clearly separating the plants from the animals.

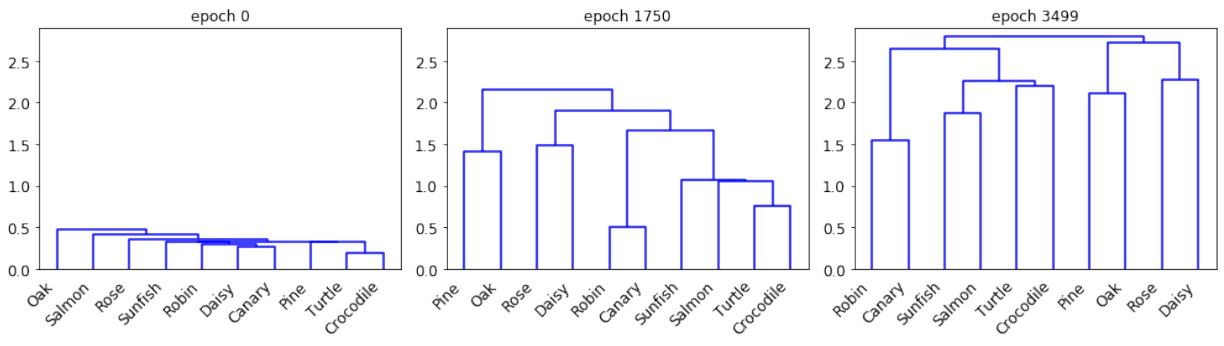

Figure 1: Dendrograms for 3 evenly spaced epochs during the training process. By the final epoch, the euclidean distances mark the network's ability to differentiate specific items.

Another interesting viewpoint of differentiation during training was by looking at the nodes directly. For similar items, activation levels of nodes typically start to hold similar values. Rather than looking at each of the 10 representation layer nodes individually, these results can be summarized through principal component analysis. By taking the learned representation layer weights after each epoch, we can map out the general trends of the activation of each node for each item. These results are summarized in Figure 2. We chose to map the PCA representation for 4 evenly spaced snapshots through the learning process, simply for visual purposes, but despite this, it is clear how similar items typically follow similar paths and end up in similar positions. It's important to note that projecting onto the first two principal components, at these 4 time points, explain around 55 65% of the explained variance in the data at a given time. Therefore, despite not being a perfect representation, the results are relatively straightforward. The pairs of Robin and Canary, and Turtle and Crocodile, for example, follow nearly identical paths according to the PCA plot, which backs the idea that these similar animals share similarities in their learned representation. The differences between dissimilar items show the differentiation over time from a multidimensional standpoint.

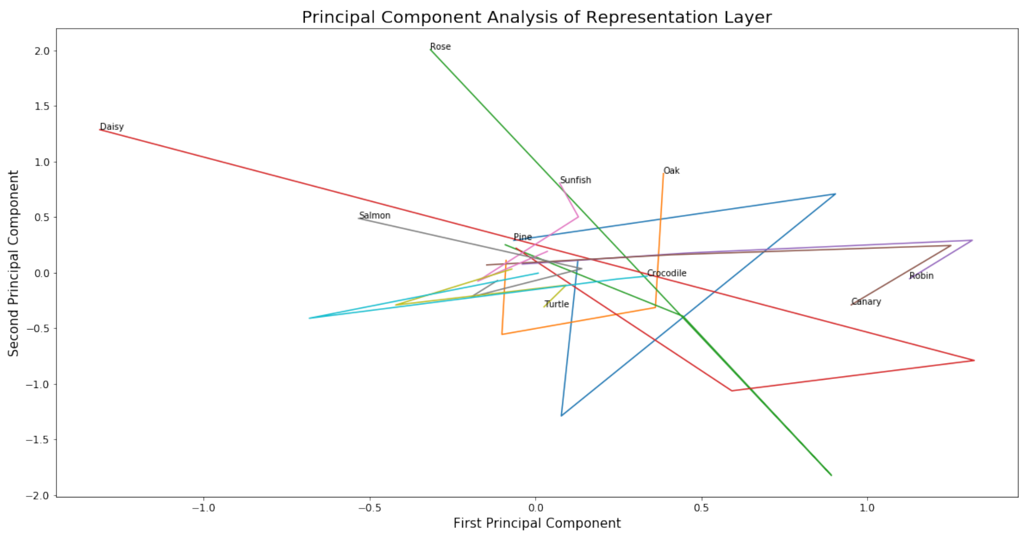

Figure 2: A basic principal component analysis of the representation layer. It shows a two-dimensional projection of the representation of the items through the learning process.

A final analysis of differentiation through learning was by looking within a given item. While Figures 1 and 2 clearly outlined the differentiation between items, it's valuable to observe how attributes within an item are learned. We group attributes according to where their characteristics lay within the hierarchical structure. So for example, if looking at the Crocodile for the "Is A" relation: "Animal" is a very general characteristic, while "Reptile" is classified as an intermediate characteristic, and finally "Crocodile" is very specific. Figure 3 shows the relationship between these general, intermediate, and specific attributes as a function of time of learning. It was very clear that these general characteristics were learned much more quickly than intermediate and specific characteristics, as given by how quickly they achieve an output layer activation closer to one.

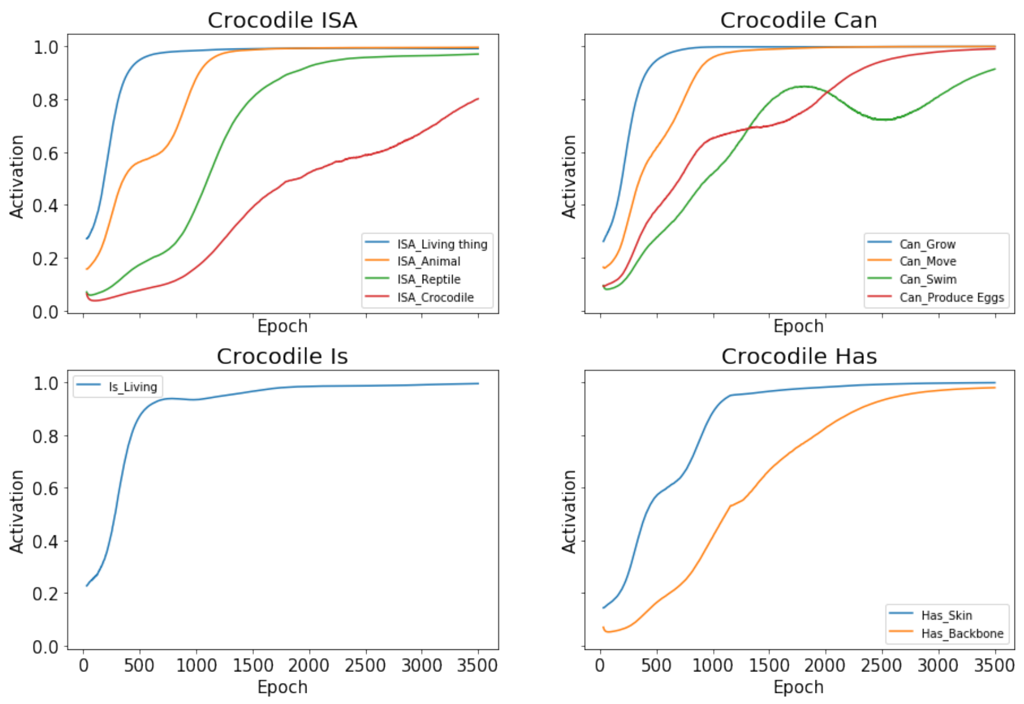

Figure 3: The figure compares the activation of the output layer for a sample item with four relations (ISA, Can, Is, Has) for the Crocodile input item. The x-axis is the number of epochs. The y-axis is the activation value in the output layer. Color lines indicate specific attributes ranging from general to specific.

4.2 Network Deterioration

Noise was implemented in two ways. The first way was by adding Gaussian noise to the representation layer. The second way was by randomly dropping out nodes in the representation layer. Model outcomes were evaluated based on positive and negative expectations. Positive expectations correspond to the correct learning between Items, Relations, and Attributes, for example *crocodile* (Item) *is a* (relation) *reptile* (attribute). Negative expectations correspond to the unrelated learning between Items, Relations, and Attributes, for example *crocodile* (Item) *is a* (relation) *bird* (attribute), or *crocodile* (Item) *has* (relation) *wings* (attribute). Noise was added as explained in the Methods section, and results are shown in Figures 4 and 5. The noise level is meant to simulate levels of neuron deterioration, and it is clear how original strong activation starts to deteriorate, while original weak activation tends to stay unchanged.

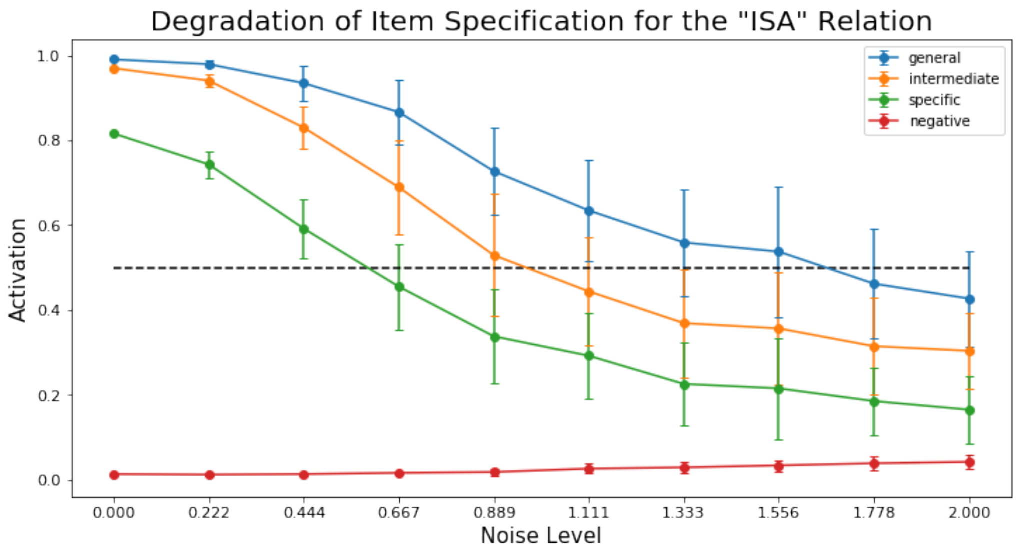

Figure 4: The x-axis shows the variance of the Gaussian distribution which noise was sampled from. The y-axis shows the activation value of the output layer. The dash line indicates a 0.5 activation value, where >= 0.5 indicates a positive result in the output. The colored lines (blue, orange, green) show the three types of attribution learning based on positive expectation. The red line shows the attribution learning based on negative expectation.

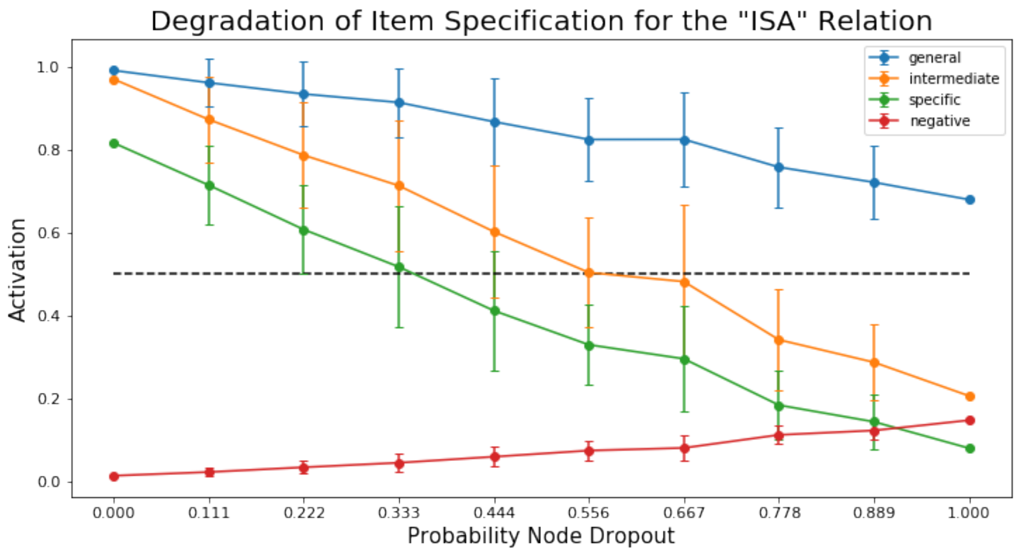

Figure 5: The x-axis shows the probability of node dropout, used to simulate level of noise. The y-axis shows the activation value of the output layer. The dash line indicates a 0.5 activation value. The colored lines (blue, orange, green) show the three types of attribution learning based on positive expectation. The red line shows the attribution learning based on negative expectation.

5 Discussion

In this study, we constructed a neural network model to study semantic cognition and how disorders like semantic dementia can be understood through introducing noise to the model. The theoretical and model framework was borrowed from McClelland and Rogers' paper [1]. Our training results replicated a similar finding, showing a graduate progression of learning of attributes ranging from general to specific characteristics. Our change to BCELoss, mini-batch gradient descent, and a learning rate of 0.05 allowed for perfect classification of multi-label attributes, and a clear distinction of item differentiation and hierarchical structures.

In our findings on differentiation through learning, it was quite clear how the general, intermediate, and specific attributes were learned over time. General attributes were learned much more quickly than specific attributes, which would be expected in nature and corresponded to the developmental progress in children's learning (i.e. learning that an item is an animal vs. learning its specific name). The top right plot of Figure 3 showed the network's relative difficulty in learning attributes that were shared between other items in training. While the Crocodile can swim, so can both the Sunfish and Salmon. While the Crocodile can produce eggs, so can the Canary and Robin. The network seemed to be somewhat perplexed by this "Swim" attribute, as shown by the drop in output layer activation value. These shared attributes were added throughout the hierarchical structure to observe difficulties in the learning process, and it is clear that the network at certain points struggled to learn these shared attributes, however, with enough time it was able to overcome this.

Our two ways of implementing noise showed a degradation of item specification as a function of increasing noise. In the Gaussian noise implementation, all three levels of attribute learning (general, intermediate, specific) showed a similar rate of degradation, and the activation for negative expectation remained low. When there was no noise implemented, general attributes had higher activation than both intermediate and specific attributes, and its activation only started to drop below 0.5 after a noise level of 1.56. Once the model started to dip below an output layer activation level of 0.5, this represented a loss of the ability to call these attributes, considering there is now a higher probability that the attribute can be negative.

In the nodes dropout implementation, learning of the general attributes showed a different rate of degradation than the intermediate and specific levels, and the activation for negative expectation increased as nodes were being dropped. When there were no nodes being dropped (simply representing the final state of the trained model), general attributes had higher activation than both intermediate and specific attributes. It was also much more robust against node drops in which the activation did not drop below 0.5. However, activation increased for negative expectation as more nodes were being dropped. It was interesting to note that when there was a 100% chance of nodes dropping out, there was still signal in the general attributes. This means that there was enough information stored in the final hidden layer to be able to classify general items. This could possibly be a caveat to this analysis, since none of this output information originates from the representation layer or input. Despite this, the trends were clear that the degradation in the network led to weaker signal in known positive attributes and increased signal in negative attributes.

Our Gaussian noise implementation failed to capture a similar result when compared to the original paper, but nodes dropout implementation did. Although both models captured a similar memory loss from general to specific when compared to semantic dementia patients, the second implementation using nodes dropout was able to reflect the robustness of the general category and explain incorrect naming of objects in dementia patients.

Another note is that in the original paper, all bias levels were initialized to -2 and were untrainable. In our model, the representation layer was forced to have no bias, but the hidden and output layers were allowed to have trainable bias. In the McClelland paper, they were able to have Gaussian random noise levels all the way up to 9, while in ours we only went to 2 before seeing complete loss of general attribute detection. The bias could be responsible for this, since their untrained bias values would make it more robust when adding noise to the weights.

Semantic dementia patients often have the ability to copy a picture accurately while it remained in view, for example, copying an image of a swan with a swan picture. However, when they are asked to reproduce the image with a delay, they have difficulty in reproducing the distinctive but not the general properties of the object [1]. In this study, dropping out the entire nodes in the representation layers did not remove the most general attributes of items and relations. The key distinctions or general properties were generally preserved. However, in the Gaussian noise model, the loss of general properties was at a higher risk as noise increases. Thus, the node dropout model captured better the distinctive memory loss of properties in the dementia patients.

While the level of noise certainly could've been increased in the visualization in Figure 4, the level of noise in Figure 5 could not have been increased any further, and it is still capable of maintaining general attribute detection. At a noise level from 1-1.6 in Figure 4, intermediate and specific attributes are lost, but general attributes remain intact, so while the dropout method seems more robust to general attributes, it could just be a matter of fine-tuning the range of noise levels in the random noise method to align with what truly occurs in people with dementia.

Semantic dementia patients sometimes misname an item, for example, saying that a rooster is a dog [1]. While there is not enough "negative" signal in either network deterioration setup to back this point (negative signals are not suddenly achieving enough signal to be considered positives), it is quite possible that increasing the complexity of the data, such as just adding more input items and/or increasing the number of attributes, could make the model more sensitive to these failures. An additional experiment conducted in the paper [1] was training with noise, which could mimic how neurons actually behave. We did not implement this, but it could very well lead to the effects just described.

In summary, adding a Gaussian noise corresponded to subtle to extreme deterioration of neurons, while node dropout suggests complete loss of function of some connection of neurons. Overall, our study was able to replicate the key findings in the McClelland & Rogers' work on semantic cognition, while slightly increasing the domain and adding additional underlying complexities to the data. While introducing noise to a neural model can generally produce the effect of memory loss from general to specific, the way in which it was implemented produced a subtle difference. We compared two ways of noise implementation and concluded the node dropouts implementation was better at capturing the loss of memory behavior in semantic dementia.

References

[1] James L. McClelland and Timothy T. Rogers. The parallel distributed processing approach to semantic cognition. Nature Publishing Group.

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[3] M.R. Quillian. *Semantic Information Processing.* MIT Press, Cambridge, Massachusetts, 1968.