An Unsupervised Learning Method for Representing Simple Sentences

Derek Monner and James A. Reggia

***Abstract*— A recent neurocomputational study showed that it is possible for a model of the language areas of the brain (Wernicke's area, Broca's area, etc.) to learn to process words correctly [1]. This model is unique in that it is a neuroanatomically based model of word learning derived from the Wernicke-Lichtheim-Geschwind theory of language processing. For example, when subjected to simulated focal damage, the model breaks down in ways reminiscent of the classic aphasias. While such results are intriguing, this previous work was limited to processing only single words: nouns corresponding to concrete objects. Here we take the first steps towards generalizing the methods used in this earlier model to work with full sentences instead of isolated words. We gauge the richness of the neural representations that emerge during purely unsupervised learning in several ways. For example, using a separate "recognition network", we demonstrate that the model's encoding of sentences is adequate to permit subsequent extraction of a symbolic, hierarchical representation of sentence meaning. Although our results are encouraging, substantial further work will be needed to create a large-scale model of the human cortical network for language.**

I. Introduction

Researchers have been investigating language learning in artificial neural networks for more than 20 years. Despite a recent abundance of research on the neuroanatomical correlates of language [2], [3], [4], a biologically plausible model of natural language learning that can be demonstrated to work at a level comparable to that of the human language learning faculty has failed to emerge. Towards that end, this paper describes some first steps towards a neuroanatomically based computational model that can learn representations of natural language at the sentence level. The model's input, a stream of phonemes, each represented by a vector of distinctive features, approximates the input to the human language learning system. The model makes sense of this stream through unsupervised learning, using only methods that are suspected to occur in biological cortex [5]. Our goal is to show that unsupervised self-organization is sufficient to shape a language reception pathway, causing it to produce distributed representations that are rich enough to support the complexities of sentence-length utterances. We employ several methods to assess the richness of the produced representations, including the use of a separate recognition network that reads from the model's output and maps its discovered representations to symbolic meanings of heard sentences.

Derek Monner and James A. Reggia are with the Department of Computer Science, University of Maryland - College Park, Maryland, USA (email: {dmonner, reggia}@cs.umd.edu)

This work was supported by NIH Award NS35460 and in part by NSF IGERT Award DGE-0801465.

This work will appear in the Proceedings of the IJCNN 2009 (in press)

Previous work by Elman [6] seems to capture some of the important characteristics of human language acquisition. Elman trained a simple recurrent network using error backpropagation [7] on a next-letter prediction task, using a large corpus of text as input. After the network had become adept at the prediction task, characteristic representations for the words in the training set were determined by averaging the network's activations following many in-context presentations of each word. A hierarchical clustering analysis of these representations showed that words with similar functions were grouped together in the hierarchy. Further, the network was able to correctly place novel words in the hierarchy based solely upon where they occurred in context and without additional training. While this work was revolutionary and influential, it suffers from a lack of biological plausibility due to the use of error backpropagation [8] as the sole learning method. The absence of local lateral interactions between nodes, which are known to occur in the human cerebral cortex, further weakens the work's biological analogy. Our model produces similar clustering results via a form of completely unsupervised learning which seems to be a more biologically plausible method [5].

Our model is based on a modified one-shot multi-winner self-organizing map [5], a neural network that learns in an unsupervised fashion to aggregate input sequences into distributed activation patterns. Earlier work established a set of parameter values (many of which we borrow directly) suitable for learning to produce unique activation patterns for differing sequences of phonemes representing single words. This past work also demonstrated that a trained layer can become a topologically clustered map with respect to its phoneme inputs [5], and that mirror-image topographical maps also occur naturally in the model [9], which provides convincing support for the analogy between this type of neural network and biological cortex.

A recent neuroanatomically grounded model of the cerebral language centers [1], based on the Wernicke-Lichtheim-Geschwind model, used one-shot multi-winner self-organizing maps to great effect. This model learned to associate 50 concrete nouns (presented as phoneme sequences) with representative visual images, such that it could repeat heard words and provide the name of seen pictures. After artificial "lesioning" or damaging of various pieces of the model, it was able to reproduce deficits that closely mimicked the classic aphasia syndromes in human subjects, exposing a deep connection between the model and human language processing. However, this work was, by design, limited to processing single words. Our work directly extends the language recognition portion of this past model by expanding the phoneme sequence input from single

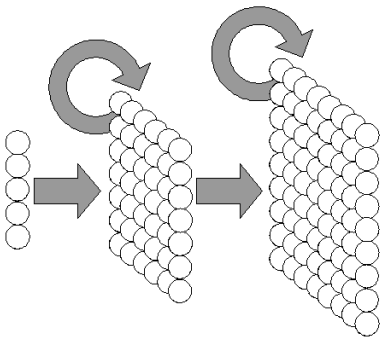

Fig. 1. The information flow in our recurrent neural network model representing the early stages of cortical language reception. Circles represent neural processing units, though the number of elements per layer is reduced in this diagram for ease of presentation. Straight arrows represent full forward connectivity, and circular arrows represent local recurrent connectivity between neighboring nodes within a layer. The input layer (left) is a vector of binary units onto which the feature vector representations of individual phonemes are imposed sequentially, representing a heard sentence. After the presentation of each phoneme, activation propagates to the *word layer* (middle), which learns distributed representations for the various phoneme combinations that comprise words by modifying the local lateral and afferent weights in an unsupervised fashion. The *sentence layer* (right) receives input from the word layer, and functions in much the same way, but instead learns distributed representations of entire sentences.

words to structured multi-word utterances, for the first time challenging a neuroanatomically based model to learn to process complete sentences.

II. Methods

Our model is designed to take as input a temporal sequence of phonemes representing a sentence and produce a representation of that sentence distributed over its output layer. Our hypothesis is that each learned distributed representation will be rich enough to allow recovery of the meaning of the associated sentence. Furthermore, the trained layers of the model should develop organizational properties that, when examined, attest to the linguistic knowledge the model has acquired.

The following subsections will describe the model's recurrent architecture, activation dynamics, and unsupervised learning methods in detail, before taking a brief detour into the processes we used to generate meanings and sentences to use as targets and inputs, respectively. We then discuss several techniques for gauging model performance in the experimental methods section.

A. Model Architecture and Dynamics

Information flow in the model (see Fig. 1) begins in the input layer, a vector of binary neural elements upon which input phonemes are imposed in sequence. Each phoneme is represented as a 34-bit feature vector, following previous work [1], [5]. For space reasons, we omit the full list of phonemes and associated feature vectors, but the interested reader can find this information in Fig. 10 of [1]. After a phoneme is presented to the input layer, activity propagates via the weighted connections to the *word layer*, a special type of self-recurrent layer, which will be discussed below. During training, the word layer learns immediately after it is activated and resets its activity values to zero after it has received a complete word. This process repeats as phonemes are presented, until, following word layer activation on the last phoneme of a word, activity propagates to the *sentence layer*. The sentence layer is of the same type as the word layer, though typically larger, as its task of aggregating words into sentences is significantly more difficult than aggregating phonemes into words. The sentence layer also learns after each activation and, after witnessing a complete sentence, clears its activity before processing the next sentence. The clearing of these latter two layers effectively lets the model know word and sentence boundaries *a priori*. This design choice was made for ease of analysis—it allows us to segment the distributed representations on each layer and associate them with specific words and sentences.

The word layer and sentence layer are modified versions of one-shot multi-winner self-organizing maps [5]. This type of layer is a hybrid descendent of Kohonen's self-organizing maps [10] and the more realistic but computationally expensive models of von der Malsburg [11]. In essence, these layers are Kohonen maps modified to support multiple concurrent local winners. They are thus recurrent neural networks that learn via competitive, unsupervised Hebbian training rules (described below). This type of layer has been found to mimic some organizational properties of biological cortex [1], [9].

Each layer is a two-dimensional $m \times n$ grid of neural elements, with each node i maintaining a real-valued non-negative activation level a_i. Each node within the layer has a weighted afferent connection to each node in the previous layer; collectively these afferent weights are represented by the matrix A. Each node is also connected laterally to other nearby nodes in the same layer to form a local neighborhood. The set of neighbors of node i is denoted N_i and defined based on a distance metric d and a radius r, as follows:

$$N_i = \{\text{nodes } j \text{ s.t. } d(i,j) < r\} \tag{1}$$

The distance function d above is a box-distance metric that treats the layer as a toroidal surface in which the left and right edges connect seamlessly, as do the top and bottom. Intuitively, $d(i,j)$ is the radius of the smallest box, centered at i, that includes j. If nodes i and j are located at row and column (r_i, c_i) and (r_j, c_j) respectively, $d(i,j)$ is:

$$d(i,j) = \max \begin{cases} \min \begin{cases} |r_j - r_i| \\ m - |r_j - r_i| \end{cases} \\ \min \begin{cases} |c_j - c_i| \\ n - |c_j - c_i| \end{cases} \end{cases} \tag{2}$$

Neighboring nodes share activation from one time step to the next via weighted connections, represented by the (sparse) matrix L. In any given time step, neighbors also compete with each other for activation, as follows: The net input to a node i at time t, denoted $I_i(t)$, is calculated as the weighted sum of its inputs from afferent and lateral sources. Any node

that has a net input higher than all of the others in its connection neighborhood is dubbed a "winner". We let W denote the set of winning nodes. Each non-winning node has its activation reset to zero, and then receives some small amount of activation from each winner in its neighborhood, although the amount of this activation decreases exponentially with the node's distance from the winner. Denoting the activations of the input units to the layer at time t by the vector $\vec{b}(t)$, we can express activation rules as:

$$I_i(t) = \alpha \, \vec{A}_i \cdot \vec{b}(t) + (1 - \alpha) \, \vec{L}_i \cdot \vec{a}(t-1) \tag{3}$$

$$W = \{\text{nodes } i \text{ s.t. } I_i(t) > I_j(t), \, \forall j \in N_i\} \tag{4}$$

$$a_i(t) = \begin{cases} I_i(t) & \text{if } i \in W \\ \sum\limits_{j \in N_i \cap W} I_j(t) \, \gamma^{d(i,j)} & \text{otherwise} \end{cases} \tag{5}$$

where γ represents the activation dropoff factor, and α is a tuning factor for the relative strength of afferent versus lateral connections; both of these are constants in the interval $[0, 1]$.

After each round of activation during training, the layer learns in an unsupervised fashion. Afferent weights associated with connections between layers are updated according to the Hebbian rule:

$$A_{ij} \leftarrow A_{ij} + \mu \, a_i(t) \, b_j(t) \tag{6}$$

where μ is the afferent learning rate. Lateral weights within each layer learn using a temporally asymmetric Hebbian rule, which strengthens each connection in proportion to how much the sending node contributed to the receiving node's increase in activation between the previous time step and the current one:

$$L_{ij} \leftarrow L_{ij} + \eta \, a_j(t-1) \max\{a_i(t) - a_i(t-1), \, 0\} \tag{7}$$

where η is the learning rate for the lateral weights. If a node's activity did not increase since the last step, its incoming lateral weights are not updated. Note that this rule does not make much sense when $i = j$, as this weight will increase more than any other and eventually dominate the weight set. To remedy this, we force all nodes to have a fixed self-connection weight of β, which is a constant in the interval $[-1, 1]$. Both the afferent and lateral weight vectors into each layer node are normalized to unit length following each learning step in order to foster further competition among the weights and prevent unbounded weight increases.

Our self-organizing map differs from previous iterations [1], [5] in two important respects. First, the neighborhoods are not truncated at the map edges, which prevents nodes near the edges from having a lopsided connection neighborhood which could cause strange dynamics. Second, winning nodes keep their activation level instead of being fully activated, which helps to maintain differences between activation patterns in which many or all winning nodes coincide.

Thus, as the model receives sentences as input, it gradually modifies its representations of words and sentences, distributed over the word layer and sentence layer respectively. But how do we know if the new representations are better or worse in some sense than the old ones? In our experiments we employ a variety of techniques to verify the richness of the representations, but the most important indicator is the extent to which we can recover the symbolic meanings of the sentences from the learned representations. To attempt this, we use another, simpler neural network, termed the recognition network, which is described in the next section.

B. The Recognition Network

After training the model, we can use a *recognition network* to recover each sentence's symbolic meaning from the distributed representation over the sentence layer. The recognition network is not part of our language reception model; it simply provides a mechanism for determining the extent to which the model has learned to represent a unique meaning for each individual input sentence. If the recognition network can successfully recover sentence meanings, we can be reasonably certain that the distributed representations generated by the model contain enough information to "understand" the sentences. We contend that, if a simple supervised neural network can perform this translation, so could a higher brain region that is concerned with the semantic content of the heard sentence, even though the methods of learning may be different. Thus, if the recognition network is successful at its task, we know that the sentence representations learned by the sentence layer are sufficiently rich to be useful to another neural system.

The recognition network is a two-layer feed-forward neural network. Its input is the distributed representation of a sentence formed on the model's sentence layer, and its output is a symbolic meaning associated with the original sentence. All units in both layers have a logistic activation function, and weight updates are performed by RPROP [12], using the normal error function on the hidden layer; the output layer is trained with an asymmetric error function designed to bias towards active units, since active units are expected to be sparse in correct output.

The form of the symbolic meanings that the recognition network attempts to recover is described in the next section, which also explains how we generate input to train the model.

C. Meaning and Syntax

Any language learner, whether natural or artificial, will encounter a finite set of meanings in its lifetime. For our experiments we chose to carefully define the *meaning space*, or the set of meanings that the model will attempt to learn. The meaning space used in our experiments is focused on a simple tabletop environment, in which several different kinds of objects reside on a small, flat surface. This meaning space is capable of describing these objects, as well as simple actions, such as grabbing, moving, and placing, that involve the objects. We chose this setting for our meaning space for possible future applicability with an embodied model. Language experiments in the human aphasia literature often involve working with objects in such an environment; use of this meaning space provides important comparison opportunities between that future model and the existing literature.

```
Meaning(
  type=put,
  obj=Object(
    det=Determiner(d=definite),
    shp=Shape(s=block)
  ),
  loc=Location(
    rel=Relational(r=near),
    det=Determiner(d=definite),
    col=Color(c=red),
    shp=Shape(s=pyramid)
  )
)
```

Fig. 2. An example of the hierarchical representation of a specific meaning that is legal in the meaning space defined in Appendix I. This meaning is for the sentence "put the block near the red pyramid".

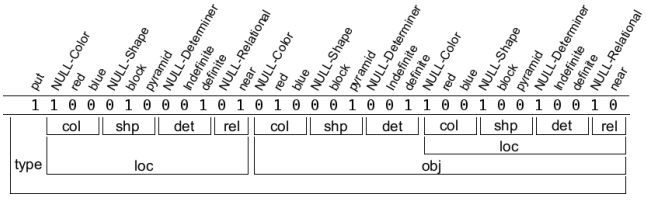

Fig. 3. An example meaning vector corresponding to the meaning in Fig. 2 ("put the block near the red pyramid"). Below the vector is the hierarchical grouping of the bits into the fields described by the meaning space. Each such group has a null bit that is active if no other bits in the group are active. Since our recognition network will try to generate meaning vectors, the null bits allow it to use a winner-take-all approach for each group of bits, since each group will always have exactly one bit active. Note that the depicted meaning vector is simpler than the one used in our experiments, as some superfluous bits have been removed for space considerations.

The meaning space is comprised of rules defining *meaning predicates* and *meaning atoms*, a full list of which can be found in Appendix I. A fully instantiated example meaning is shown in Fig. 2.

The goal of the recognition network is to recover these meanings when given an English sentence—a temporal sequence of phonemes—as input. We are aware of no neural network models that are able to learn output of an *explicitly* hierarchical nature. As such, we take the approach of converting each hierarchical meaning into a one-dimensional binary vector, termed a *meaning vector*, which the recognition network attempts to learn from the distributed sentence representations created by the model. We define a bijection between meanings and meaning vectors such that any meaning legal in the meaning space can be represented by a meaning vector, and we can easily recover the hierarchical meaning given only the flat meaning vector.

Meaning vectors are constructed such that each bit represents a single meaning atom coupled with a position at which it may appear in the meaning hierarchy. Meaning vectors contain extra null bits, one of which is activated when a specific field in a meaning is not instantiated; these are useful when performing winner-takes-all dynamics on groups of related meaning atoms. Fig. 3 shows the example sentence from Fig. 2 in meaning vector form.

In our experiments, we randomly select a set of meanings from the meaning space to use to create natural language input data for training the model. To turn these hierarchical meanings into natural language fit for consumption by our model—or a human—we use a set of syntactic rules that collectively produce a grammatical English sentence from a meaning. The rules define mappings to be used, subject to certain conditions, to transform a given meaning predicate or meaning atom into a sequence of words in natural language. The words are then broken down into their constituent phonemes and concatenated into a single uninterrupted sequence. Any phonetically expressible language could be used, but we chose English for our experiments. The full set of syntax rules used in our experiments is in Appendix II.

TABLE I

MODEL PARAMETER VALUES USED IN EXPERIMENTS.

Parameter	Word Layer Value	Sentence Layer Value
$m \times n$	20×20	30×30
r	3	
α	0.2	
β	0.5	
γ	$0.37/(1+e^{(t-0.2)/0.16})$	
μ	$0.4/(1+e^{(t-0.44)/0.0001})$	
η	$0.62/(1+e^{(t-0.3)/0.04})$	

Once a meaning has become a sequence of English words, we use the CMU Pronouncing Dictionary [13] to convert each word into its constituent phonemes. We represent each phoneme by a 34-bit feature vector as discussed in section II-A. The phonemes from all the words and sentences are concatenated to form one long phonemic sequence, to be fed to our model one phoneme at a time.

D. Experimental Methods

1) Map Formation: We first performed a set of experiments designed to gauge the effectiveness of map formation on both the word layer and sentence layer. We used a simple meaning space containing only command-type sentences, with 4 verbs, 6 nouns, 5 adjectives, 5 prepositions, and 2 articles (see Appendix I). This meaning space can generate over 50,000 legal meanings. From this meaning space we generated 500 unique, random meanings and their associated sentences to form the training set. Crucially, we made sure that no meaning in the training set contained any of 3 specific meaning atoms. After training, we use sentences generated using any of the 3 novel meaning atoms to assess the ability of the model to encode unfamiliar input words.

Table I shows the model parameters used in the experiments reported here, which were obtained from pilot experiments. Parameters γ, μ, and η are determined by functions that decrease monotonically as training progresses, following [5]. In those functions, $t \in [0, 1]$ indicates the amount of training complete, with $t = 0$ indicating the beginning of training and $t = 1$ the end of training.

We trained the model for 100 epochs, presenting all 500 sentences from the training set in random order during each epoch. During training, we tracked map formation by

recording the activation patterns present on the word layer after the presentation of each word, as well as those on the sentence layer following each sentence. Ideally, the maps should learn a distinct activation pattern for each word and sentence to ensure the uniqueness of the learned meanings, and should also spread these patterns out in space to make them easier to diffentiate from each other. Our first instinct was to use Euclidean distance to measure how spread out the various patterns were, but this technique requires treating the 2-dimensional layer as a 1-dimensional vector for the comparison, so adjacency information is lost. Instead, we analyzed this using a metric we call *winner separation*, which captures how closely the set of winning nodes for one activation pattern resembles the winners of another pattern, as laid out in space on the 2-dimensional map. If W_1 is the set of winning nodes for one pattern, and W_2 contains the winners of another pattern, then the winner separation is:

$$d_{win}(W_1, W_2) = \sum_{i \in W_1} \min_{j \in W_2} \{d_{box}(i,j)\} + \sum_{j \in W_2} \min_{i \in W_1} \{d_{box}(i,j)\} \quad (8)$$

This metric is appropriate in that patterns with identical winner sets will score a distance of zero, and distance increases linearly as the patterns begin to have fewer winning nodes in common and those nodes become further from each other in the grid. Measuring the mean winner separation across all pairwise combinations of patterns gives us a good idea whether there is a general trend towards spreading out patterns.

2) Meaning Recovery: Our second set of experiments tests the richness of the distributed representations learned by the model by attempting to recover the meanings directly using the recognition network. The model parameters are still as in Table I. We use the same meaning space, generating 500 unique sentences, this time using all meaning atoms in the training set.

For these experiments, the recognition network described in Section II-B is instantiated with 100 hidden units and 65 output units, the latter number corresponding to the size of the meaning vector for our meaning space. After model training has completed, we train the recognition network using the RPROP algorithm. Borrowing notation from [12], we used parameters $\Delta_0 = 0.1$, $\eta^+ = 1.2$, and $\eta^- = 0.5$, for the initial per-connection learning rate, learning rate growth factor, and learning rate decay factor, respectively.

After training the recognition network, we assess performance in two ways—by measuring the number of meaning atoms recovered correctly versus those omitted or spuriously generated; and by counting the number of full meaning vectors that the recognition network correctly recovers from each set. The phrase "correctly recovers" deserves some explanation. We apply winner-takes-all activation logic to the output produced by the recognition network for a given input sentence; that is, we compare the activations of all units in each group of related meaning atoms in the produced meaning vector, and the winner in each group becomes fully active, while the others are turned off completely. This creates a binary vector from the recognition network's output, which is then compared with the target meaning vector. If and only if the two match exactly, the recognition network scores a correct recovery. The percentage of meaning vectors correctly recovered in this way is termed the *accuracy* of the network. These two measures together provide a view of the performance of the recognition network that is easier to understand than standard error metrics such as MSE.

III. Results

A. Map Formation

We performed 20 independent trials of the map formation experiment. Fig. 4 shows the mean winner separation between each pair of patterns as training progresses, for both the word layer and the sentence layer, averaged over these 20 trials. The nearly monotonic increase in winner separation shows that the model learned to separate its representations from each other over time, making them easier to distinguish. The winner separation on the word layer levels out before the half-way point in training, probably due to the relatively small number of patterns it has to learn: 22 word patterns, versus 500 sentence patterns for the sentence layer.

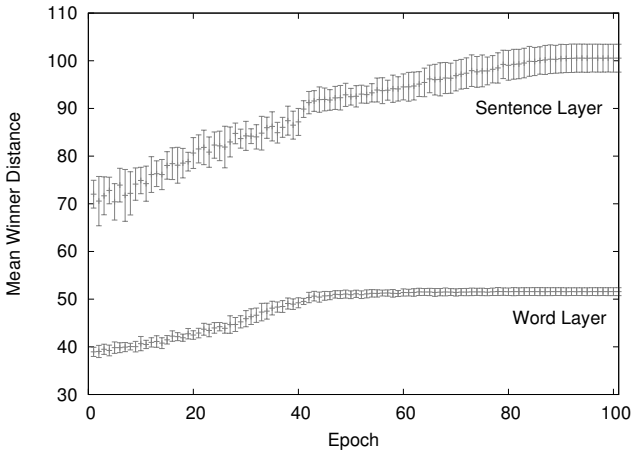

Fig. 4. The mean winner separation at each epoch, calculated as the average winner separation between all pairs of words (for the word layer) or sentences (for the sentence layer) in the training set. Note that the sentence layer has a larger surface, which accounts for its higher winner separation values. Each data point represents the average over 20 repetitions of the unsupervised map formation experiment; error bars show one standard deviation.

We also examined the learned representations topographically. Fig. 5 shows a representative word layer after training. Each map unit is labeled with the input phoneme to which it responds most strongly, and the units are color-coded to represent different categories of inputs—here vowels and consonants. The maps shows a clear tendency to cluster vowels together into "islands" in a "sea" of consonants, in a way reminiscent of the centerpiece figure from [5]. Fig. 6 is a similar depiction of the sentence layer, using the word layer's learned representation of each word as the input patterns. This image shows something even more interesting—the

sentence layer naturally learns to cluster the input words by their function, even though its only clue is the temporal occurrence patterns of the words in the input.

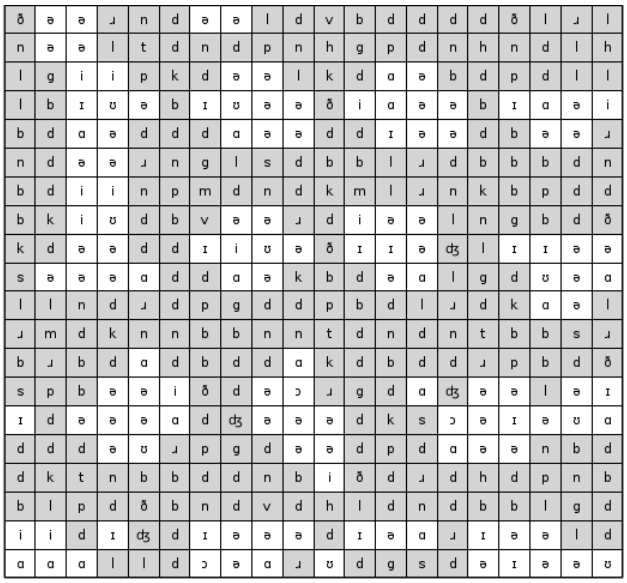

Fig. 5. A view of the trained word layer where each unit is labeled with the input phoneme that activates it most strongly. The units are also shaded by type of phoneme, with vowels in white and consonants in gray. The clustering of the vowels suggests that the map learned to differentiate the two types of phonemes based upon their patterns of occurrence within the input.

By watching the sequences of words as they pass by, the sentence layer seems to learn that some words are more-or-less interchangeable syntactically, and groups these words together. To gather further evidence of these groupings, we ensured that the training set excluded the noun "pyramid", the adjective "orange", and the preposition "below". After training was complete, the model was tested on novel sentences from the original meaning space, some of which contained one or more of these three omitted words. Following Elman [6], we performed a hierarchical clustering analysis on the in-context activation patterns on the sentence layer after the presentation of each word, averaged over each occurrence of the word. As can be seen in Fig. 7, after training, the model appears to classify novel words that appear in familiar contexts and group these words with their learned counterparts, despite having never been trained on these words.

B. Meaning Recovery

The results from the previous set of experiments suggest that our model learns to create distributed representations for the input sentences that, at least to some extent, take into account the relationships between words. In order to quantify how rich these representations are, we attempted to map each to the hierarchical meaning underlying the associated sentence by training the recognition network after model training was complete.

Fig. 8 shows, on average, how many meaning atoms the recognition network should recover for each sentence, versus

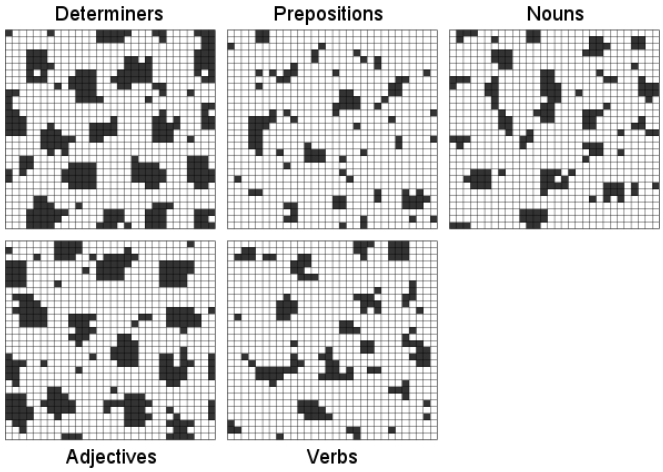

Fig. 6. We looked at a representative trained sentence layer and noted, for each unit, the input word that caused the strongest response in that unit. The units were then partitioned based on the syntactic category of that word, resulting in the five separate diagrams above, each representing a different word category. The clear clustering effect shows that the map has learned to group words by their function, as determined by their temporal occurrence patterns.

how many of those the network recovered correctly. It also shows how many omissions and spurious generations the network made. The recognition network does very well, asymptotically approaching full recovery and virtually eliminating omissions and spurious generations. Finally, Fig. 9 shows the accuracy scores of the recognition network. We can see that the recognition network was able to perfectly recover the meaning of, on average, approximately 95% of the sentences on which the model was trained. These results together show that the model's learned sentence representations are sufficient for recovering the meanings of sentences it has seen before.

IV. Discussion & Future Work

The results discussed in the previous section show that our model was able to learn something about the phonemes, words, and sentences it received as input, using purely unsupervised learning methods on a neuroanatomically grounded network model. The word and sentence representations it generated were unique and spread out in space, increasing their discriminability. The trained word and sentence layers were topologically organized into clusters of units based on the functional categories of the units' preferred inputs. The learned representations of words were clustered hierarchically by function, and the model's spontaneous representations for novel words fit neatly into this hierarchy. These model results represent predictions about the organization of human language cortex that will hopefully be testable in the near future using non-invasive methods. Finally, the learned sentence representations proved to be rich enough for the recognition network to recover hierarchical meanings of sentences the model had previously encountered. Thus, we believe that the language reception model described in this paper provides a convincing, biologically plausible

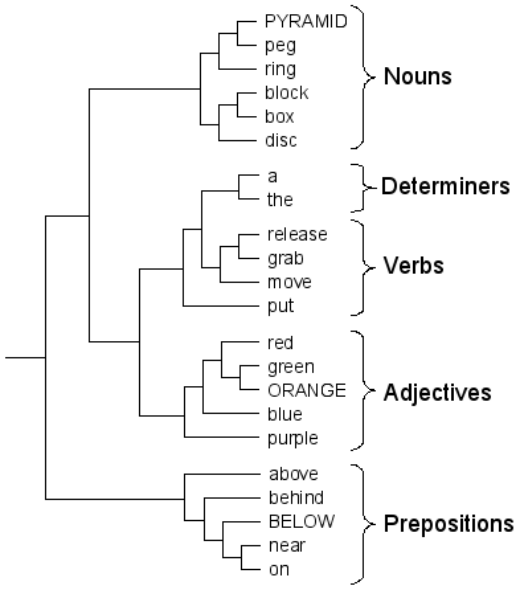

Fig. 7. A hierarchical clustering analysis of the in-context activation patterns on a typical sentence layer following the presentation of each word. Capitalized words were not present during training. This clustering suggests that the model has not only partitioned the words into classes, but that it can effectively classify novel stimuli based on their patterns of occurrence in sentences.

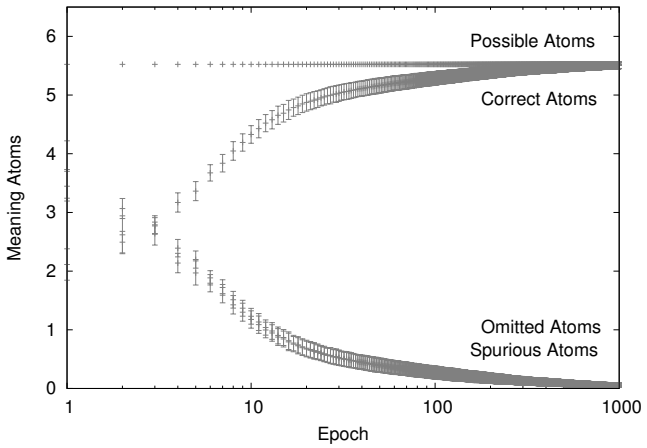

Fig. 8. With training, the recognition network is able to recover nearly all meaning atoms while hardly generating any spurious ones.

beginning for natural language understanding at the sentence level. Though it does not yet function at a level remotely comparable to the human language faculty, we believe it is an important step in the right direction.

Generalization is an extremely important aspect of generative language learning that we hardly touched on in this work. The first priority for future work with this model is a study of how a recognition network performs at recovering sentences that the model has not seen before, but nonetheless come from the same meaning space as the set of sentences on which the model was trained. After training, the model should ideally be able to generate unique representations for these related-but-novel sentences such that the trained

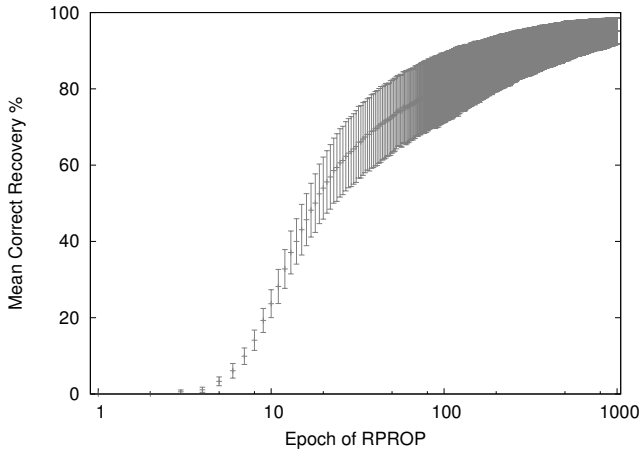

Fig. 9. The accuracy (correct recovery percentage) of the recognition network during training. Each point is averaged over 20 trials, with error bars representing standard deviation. The average accuracy approaches $478/500$ sentences (95.6%).

recognition network could recover their meanings without error. Such a result would confirm that the model had, in a sense, generalized to the entire meaning space, which is what normal human language learners are observed to do.

The model presented here is the first piece of a larger model that attempts to learn natural language at the sentence level while retaining biological plausibility. This involves the reception pathway (discussed herein), a cognitive facility for manipulating meanings gleaned from the input pathway, and a production pathway to allow the model to "speak". After such a model is completed and successfully trained, we intend to study the effects of "lesioning" the model [1]. The data obtained from these tests will be directly comparable to data from human aphasia patients, which will either validate the model or direct further research in modifying it to more closely mimic the behavior of the human language faculty.

REFERENCES

[1] S. Weems and J. Reggia, "Simulating single word processing in the classic aphasia syndromes based on the Wernicke–Lichtheim–Geschwind theory," *Brain and Language*, vol. 98, no. 3, pp. 291–309, 2006.

[2] D. Poeppel and G. Hickok, "Towards a new functional anatomy of language," *Cognition*, vol. 92, no. 1-2, pp. 1 – 12, 2004.

[3] D. Caplan, "Aphasic syndromes," in *Clinical neuropsychology*, K. Heilman and E. Valenstein, Eds. New York: Oxford University Press, 2003, pp. 14–34.

[4] H. Damasio, D. Tranel, T. Grabowski, R. Adolphs, and A. Damasio, "Neural systems behind word and concept retrieval," *Cognition*, vol. 92, no. 1-2, pp. 179 – 229, 2004.

[5] R. Schulz and J. Reggia, "Temporally asymmetric learning supports sequence processing in multi-winner self-organizing maps," *Neural Computation*, vol. 16, no. 3, pp. 535–561, 2004.

[6] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[7] D. Rumelhart, G. Hinton, and R. Williams, *Learning internal representations by error propagation*. MIT Press Cambridge, MA, USA, 1986.

[8] P. Mazzoni, R. Andersen, and M. Jordan, "A more biologically plausible learning rule for neural networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 88, no. 10, pp. 4433–4437, 1991.

[9] R. Schulz and J. Reggia, "Mirror symmetric topographic maps can arise from activity-dependent synaptic changes," *Neural Computation*, vol. 17, no. 5, pp. 1059–1083, 2005.
[10] T. Kohonen, *Self-Organizing Maps*. Springer, 2001.
[11] C. von der Malsburg, "Self-organization of oriented sensitive cells in the striate cortex," *Kybernetik*, vol. 14, pp. 85–100, 1973.
[12] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *Proceedings of the IEEE International Conference on Neural Networks*, vol. 1993, pp. 586–591, 1993.
[13] R. Weide, "The Carnegie Mellon Pronouncing Dictionary [cmudict 0.6]," 1998, Carnegie Mellon University: [http://www.speech.cs.cmu.edu/cgi-bin/cmudict].

APPENDIX I
MEANING SPACE DEFINITION

In the following, each statement represents a specific definition for a meaning predicate, whose symbols start with an uppercase letter. The symbols starting with lowercase letters are field identifiers when they appear on the left side of an assignment operator (=), or meaning atoms on the right side. When instantiating a predicate, all of its fields are required by default; a question mark following a field identifier indicates optionality of that field. Brackets after a predicate name enclose a list of conditions on the instantiation of that predicate in the given context; a minus sign before a predicate name in this list means that the given predicate cannot be instantiated as part of the parent predicate. For example, given `Meaning(type=release, obj=Object[-loc])`, we will never instantiate the `loc` field of the `obj` field that is part of this `Meaning`.

The following meaning space can generate 52344 legal meanings. The meaning vector used to represent each meaning is 65 bits long.

```
Meaning(type=put, obj=Object[-loc], loc=Location)
Meaning(type=grab, obj=Object)
Meaning(type=release, obj=Object[-loc])
Meaning(type=go, loc=Location)

Object(det=Determiner, col?=Color,
  shp=Shape, loc?=Location)

Location(rel=Relational, det=Determiner,
  col?=Color, shp=Shape)

Determiner(d=indefinite)
Determiner(d=definite)

Color(c=red)
Color(c=blue)
Color(c=green)
Color(c=purple)
Color(c=orange)

Shape(s=block)
Shape(s=box)
Shape(s=disc)
Shape(s=peg)
Shape(s=pyramid)
Shape(s=ring)

Relational(r=above)
Relational(r=below)
Relational(r=behind)
Relational(r=near)
Relational(r=supported_by)
```

APPENDIX II
SYNTAX RULE DEFINITIONS

Each line below represents a syntax rule that transforms the indicated meaning predicate into a string of words, possibly invoking other rules along the way. The bracketed items after the predicate name are conditions that must be true in order for the rule to apply. When trying to apply a rule to a given predicate, the first applicable rule in the list is always chosen and any remaining rules are ignored. Tokens in quotes on the right-hand side of the rule are English words or phrases that make up the final generated sentence. Other tokens on the right-hand side represent the names of fields in the meaning predicate being expanded that themselves need to be recursively expanded in place by applying the appropriate syntax rule for that field. After all applicable rules have been applied, only quoted words will remain, which are then concatenated in order to form the natural language sentence produced by the rule.

The following syntax rules are sufficient to convert any meaning generated by the meaning space listed in Appendix I into a valid English sentence.

```
Meaning[type==put] -> "put" obj loc
Meaning[type==grab] -> "grab" obj
Meaning[type==release] -> "release" obj
Meaning[type==go] -> "move" loc

Object -> det col shp loc

Location -> rel det col shp

Determiner[d==indefinite] -> "a"
Determiner[d==definite] -> "the"

Color[c==red] -> "red"
Color[c==blue] -> "blue"
Color[c==green] -> "green"
Color[c==purple] -> "purple"
Color[c==orange] -> "orange"

Shape[s==block] -> "block"
Shape[s==box] -> "box"
Shape[s==disc] -> "disc"
Shape[s==peg] -> "peg"
Shape[s==pyramid] -> "pyramid"
Shape[s==ring] -> "ring"

Relational[r==above] -> "above"
Relational[r==below] -> "below"
Relational[r==behind] -> "behind"
Relational[r==near] -> "near"
Relational[r==supported_by] -> "on"
```