Biologically Plausible, Human-scale Knowledge Representation

Eric Crawford (e2crawfo@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON, N2L 3G1

Matthew Gingerich (majugi@cs.ubc.ca)
University of British Columbia, Vancouver, BC, V6T 1Z4

Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON, N2L 3G1

Abstract

Several approaches to implementing symbol-like representations in neurally plausible models have been proposed. These approaches include binding through synchrony (Shastri & Ajjanagadde, 1993), mesh binding (van Der Velde & de Kamps, 2006), and conjunctive binding (Smolensky, 1990; Plate, 2003). Recent theoretical work has suggested that most of these methods will not scale well – that is, they cannot encode structured representations that use any of the tens of thousands of terms in the adult lexicon without making implausible resource assumptions (Stewart & Eliasmith, 2011; Eliasmith, in press). Here we present an approach that will scale appropriately, and which is based on neurally implementing a type of Vector Symbolic Architecture (VSA). Specifically, we construct a spiking neural network composed of about 2.5 million neurons that employs a VSA to encode and decode the main lexical relations in WordNet, a semantic network containing over 100,000 concepts (Fellbaum, 1998). We experimentally demonstrate the capabilities of our model by measuring its performance on three tasks which test its ability to accurately traverse the WordNet hierarchy, as well as to decode sentences employing any WordNet term while preserving the original lexical structure. We argue that these results show that our approach is uniquely well-suited to providing a biologically plausible, human-scale account of the structured representations that underwrite cognition.

Keywords: knowledge representation; biologically plausible; scaling; neural; vector symbolic

Introduction

One of the central challenges for contemporary cognitive modelling is scaling. As Jeff Hinton remarked in his address to the Cognitive Science Society, "In the Hitchhiker's Guide to the Galaxy, a fearsome intergalactic battle fleet is accidentally eaten by a small dog due to a terrible miscalculation of scale. I think that a similar fate awaits most of the models proposed by Cognitive Scientists" (Hinton, 2010). This observation can be taken as a challenge for cognitive modellers: Will the principles demonstrated in a small-scale cognitive model scale up to the complexity of a human-sized cognitive system? This scaling problem has often been thought to be a special challenge for biologically inspired approaches to cognitive modelling (Jackendoff, 2002). This is because the basic principles employed in such models often do not allow for a straightforward characterization of structured representations, despite the ubiquity of such representations in cognitive behaviour. This same concern is not as immediate for symbolicist approaches which typically take structured representations to be primitive (Anderson, 2007).

In this paper, we briefly review past connectionist approaches to addressing the problem of representing structure, and discuss recent criticisms of those approaches which suggest that they will not scale. We then present a new approach that we have developed that allows for the representation and manipulation of large-scale structured representations in anatomically and physiologically plausible models of brain function. In past work we have provided theoretical arguments suggesting that this approach will scale better than others (Stewart & Eliasmith, 2011). Here, our focus is on empirically demonstrating that claim. We do so by encoding the central structural relations in WordNet into neural representations in a spiking network. We present the results of three experiments showing that 1) this information can be decoded for arbitrary lexical items, 2) lexical hierarchies of any depth within WordNet are successfully represented, and 3) these lexical representations can be combined to represent structured sentences with the same methods.

Past Approaches

There have been many approaches to representing structure in connectionist networks. We consider three of the most successful: 1) binding through synchrony; 2) mesh binding; and 3) conjunctive binding.

The suggestion that structured cognitive representations could be constructed using binding through synchrony (Shastri & Ajjanagadde, 1993) was imported into cognitive modelling from the earlier hypothesis that feature binding in vision can be accounted for by the synchronization of neurons in visual cortex (von der Malsburg, 1981). Recently, this approach has seen a revival in the DORA architecture (Doumas et al., 2008) and its variants, which focus on representing structures for analogical reasoning. In these models, the temporal relationships between connectionist nodes are employed to represent structured relations. Structured representations (e.g. bigger(Fido, Sarah)) are constructed out of four levels of representation, where nodes in higher levels represent more complex structures via their connections to nodes in lower layers. As has been argued in more detail elsewhere, this kind of representational scheme will not scale well (Stewart & Eliasmith, 2011; Eliasmith, in press) because the number of nodes needed to support arbitrary structured representations over even small vocabularies (e.g. 6000 lexical items) is larger than the number of neurons in the brain.

Notably, this is not an issue with the use of synchrony per se, but rather with the the way binding has been mapped to network nodes. However, it has also been suggested that synchrony itself will not scale well to binding complex structures (Stewart & Eliasmith, 2011; O'Reilly & Munakata, 2000).

A different approach to structured representation has been taken by van Der Velde & de Kamps (2006) in their work on the Neural Blackboard Architecture (NBA). To avoid the exponential growth in resources, the NBA employs "neural assemblies." These assemblies are temporarily bound to particular symbols using a mesh grid of neural circuits (e.g. bind(noun1, Fido)). Larger structures are then built by binding these assemblies to roles using a gating circuit (e.g. gate(agent1, bind(noun1, Fido))). Neural assemblies that bind roles and their gated word assemblies are used to define higher level structure assemblies which can be used to represent sentential structures. The use of temporary binding in this manner significantly reduces the resource demands of this approach compared to DORA. However, as argued in Stewart & Eliasmith (2011) and demonstrated in more detail in Eliasmith (in press), just to *represent* simple sentences of the form relation(agent, theme) from a vocabulary of 60,000 terms, this approach requires about 480 cm^2 of cortex, approximately one fifth of total cortical area. This is much larger than known language areas which account for both representation and *processing* of linguistic terms. Consequently, while the NBA has improved scalability compared to DORA, it remains implausible.

The final approach we consider is the class of proposals broadly called conjunctive coding approaches, or, more recently, Vector Symbolic Architectures (VSAs; (Gayler, 2003)). In general, these approaches propose some kind of nonlinear vector operation to bind two vectors together. The earliest and perhaps best known such approach is that proposed by Smolensky (1990), which employs the tensor product as the binding operation. The model presented in this paper employs a VSA which uses circular convolution as the binding operation (after Plate (2003)). The crucial difference between using the tensor product vs. using circular convolution is that for an n-dimensional vector, a tensor binding results in an n^2-dimensional vector, whereas the circular convolution binding results in an n-dimensional vector. This computational difference results in severe scaling differences when considering possible biological implementations. In particular, tensor products scale exponentially poorly as the depth of the structure increases. For example, Eliasmith (in press) shows that encoding a sentence where lexical items have hierarchical relations of depth two or greater (e.g. Sarah isA(person isA(mammal))) will require approximately 625 cm^2 of cortex. Again, this is significantly larger than relevant language areas.

The above considerations suggest two main challenges for connectionist implementations of structured representations: lexical scaling and hierarchical scaling. Lexical scaling means having a lexicon that is as large as an adult human's vocabulary. Hierarchical scaling refers to being able to encode the depth of grammatical and lexical relations found in adult humans. Any method that claims to provide appropriate scaling will have to demonstrate success along both of these dimensions. The approach that we adopt in this work employs a neural implementation of a VSA which uses circular convolution for binding. The purpose of this paper is to demonstrate empirically that our approach successfully meets both of these challenges.

Theoretical Approach and Methods

WordNet

The target of our efforts – the human-scale knowledge base that we will be encoding – is WordNet, a manually constructed lexical database of the English language (Fellbaum, 1998). WordNet's design is intended to reflect the organization of concepts in a psychologically plausible way using a handful of common relationships. In WordNet words are divided into "synsets" or synonym sets of words that have the same meaning. Words that have multiple meanings are listed in multiple synsets, so the fundamental unit in WordNet is not a word but a word sense. Each synset is linked to other synsets by relations, of which there are several types. The two relation types that are of the most interest are hypernymy and holonymy. A hypernym of a concept is the general type of the concept (i.e. dog has the hypernym canine); a holonym of a concept is something that that concept is a part of (i.e. lock has the holonym door). These relations are explicitly encoded in the lexicon we employ. The inverse of the hypernym and holonym relations are also implicitly included in our encoding, although we do not test their extraction as this requires more complex control of signal flow that is beyond our present scope. The depiction of lexical relations found in WordNet is slightly simplified, though it is sufficient for our purposes; a complete description of the simplifications made can be found in Fellbaum (1998).

Each synset can be defined in terms of its relationships with other synsets. This means that a term such as dog can be defined as:

$$\textbf{dog} = \textbf{isA}(\textbf{canine}) \textit{ and } \textbf{partOf}(\textbf{pack}) \quad (1)$$

We will make extensive use of this type of representation in our model. We think of the relations in (1) as *belonging to* the dog synset, and pack and canine as the *targets* of the relations.

Vector Symbolic Architectures

VSAs in general provide a means for representing structured knowledge using high-dimensional vectors. This makes VSAs amenable to neural implementation using the Neural Engineering Framework, which shows how to systematically use populations of spiking neurons to represent vectors and functions thereof (Eliasmith & Anderson, 2003). In a VSA, each symbol to be represented is randomly assigned a high-dimensional vector. Two core operations are provided, each of which takes two vectors as input and returns a third.

Binding ($\circledast$) two vectors returns a third vector that is *dissimilar* to both of the original vectors. The *superposition* ($+$) of two vectors returns a third vector that is *similar* to both of the original vectors. Here we employ a close relative of Holographic Reduced Representations (Plate, 2003), a type of VSA in which binding is implemented via circular convolution, superposition is implemented via vector addition and vectors representing symbols are randomly chosen from the D-dimensional unit sphere. The circular convolution returns a vector which has the same dimension as the two input vectors, which is a significant improvement over the tensor product VSA discussed in the Past Approaches section.

We can use these operations to encode graph-like structures such as WordNet. First we fix a dimension D for our vectors (D=512 in our model). Then each WordNet synset and each relation type is assigned a random vector on the D-dimensional unit sphere called an *ID-vector*. Each synset is also assigned a second D-dimensional vector, built-up using the VSA operations, which stores the structural information about the synset. To construct this vector, for each relation belonging to a particular synset, we bind the ID-vector for the relation type to the ID-vector for the target of the relation. We then superpose the results from all the relations. The following equation demonstrates this process for the dog synset:

$$\mathbf{dog} = \mathbf{isA} \circledast \mathbf{canine_{ID}} + \mathbf{partOf} \circledast \mathbf{pack_{ID}} \tag{2}$$

where all variables on the right-hand side are ID-vectors. We have disambiguated the two vectors assigned to a synset by denoting the ID-vector with the **ID** subscript. What makes (2) useful is that **dog** preserves information about its constituents; we can use a third operation, *dereferencing*, to determine what a given vector is bound to in **dog**. The dereferencing operation is performed by binding **dog** with the inverse of the given vector. As an example, imagine we want to extract the synset that the dog synset is related to via the isA relation type. We bind **dog** with $\overline{\mathbf{isA}}$, the inverse of the **isA** vector:

$$\begin{aligned}
&\mathbf{dog} \circledast \overline{\mathbf{isA}} \\
&= (\mathbf{isA} \circledast \mathbf{canine_{ID}} + \mathbf{partOf} \circledast \mathbf{pack_{ID}}) \circledast \overline{\mathbf{isA}} \\
&= \mathbf{isA} \circledast \mathbf{canine_{ID}} \circledast \overline{\mathbf{isA}} + \mathbf{partOf} \circledast \mathbf{pack_{ID}} \circledast \overline{\mathbf{isA}} \\
&\approx \mathbf{canine_{ID}} + \mathbf{partOf} \circledast \mathbf{pack_{ID}} \circledast \overline{\mathbf{isA}}
\end{aligned} \tag{3}$$

Equation (3) shows that $\mathbf{dog} \circledast \overline{\mathbf{isA}}$ is $\mathbf{canine_{ID}}$ superposed with another vector which can effectively be regarded as noise. All that remains is to remove that noise, and we will discuss methods for doing so below.

We call vectors constructed in the manner of **dog** in (2) *semantic pointers* because they are a compressed representation of their constituents, and preserve similarity (i.e. semantic) relations in their compressed form. In addition, the dereferencing operation is similar to the dereferencing of pointers in programming languages. Semantic pointers have a wide range of uses; indeed, they are central to the Semantic Pointer Architecture (Eliasmith, in press) which was used to create Spaun, currently the world's largest functional brain model, which is able to account for several perceptual, motor and cognitive behaviours (Eliasmith et al., 2012).

Now, given a semantic pointer representing a synset, we can traverse the connections between that synset and related synsets by dereferencing its semantic pointer with the ID-vector of a relation type, obtaining a noisy version of the ID-vector of the target of the relation, as demonstrated in (3).

After this initial dereferencing, we must still determine how to remove the noise from the vector returned by the dereferencing operation. Looking again at equation (3), we see that $\mathbf{dog} \circledast \overline{\mathbf{isA}}$ is similar to $\mathbf{canine_{ID}}$ since it consists of $\mathbf{canine_{ID}}$ superposed with a noise vector, and is dissimilar to the rest of the ID-vectors ($\mathbf{pack_{ID}}$, **isA**, **partOf**) since they are related to $\mathbf{dog} \circledast \overline{\mathbf{isA}}$ via the binding operation. A cleanup memory, which returns the vector in a vocabulary which is most similar to a given input vector, is a potential solution to this denoising problem. However, we want our model to be able to traverse the full WordNet hierarchy, not just a single relation, and ID-vectors alone contain no structural information. We need to have some way to move from the ID-vector for a synset to its semantic pointer. We can perform both the denoising and mapping to semantic pointer in one step by using an associative cleanup memory rather than a pure cleanup memory. In short, we take the noisy ID-vector returned by the dereferencing operation and feed it into an associative cleanup memory mapping each synset's ID-vector to its semantic pointer, thus obtaining a clean semantic pointer which can then be used in further traversals.

Neural Representation and Computation

Thus far we have described VSAs and how they can be used to encode structural knowledge such as WordNet, but have not yet said anything of how to implement them in neurons. For this purpose we turn to the Neural Engineering Framework (NEF), a set of methods for building biologically plausible models using principles for neural representation, computation and dynamics (Eliasmith & Anderson, 2003). The central idea behind the NEF is that a group of spiking neurons can represent vectors over time, and that connections between groups of neurons can compute functions on those vectors. More precisely, a group of neurons represents any of a set of vectors, that is, a vector space. The NEF provides a set of methods for determining what the connections need to be to compute a given function on the vector space represented by a group of neurons. Suppose we wish to compute the function y = f(x), where vector space x is represented in population A, and vector space y is represented in population B. To do so, the NEF assumes that each neuron in A and B has a "preferred direction vector." The preferred direction vector is the vector (i.e. direction in the vector space) for which that neuron will fire most strongly. Consequently, the spiking activity of every neuron in a population A can be written

$$a_i(x) = G[\, \alpha_i e_i x \, + \, J_{bias} \,] \tag{4}$$

where a_i is the ith neuron in the population, G is the spiking neural nonlinearity, α_i is the gain of the neuron, e_i is the preferred direction (or encoding) vector, and J_{bias} is a bias current to account for background activity of the neuron. The elements in the square brackets determine the current flowing into the cell, which then drives the spiking of the chosen single cell model G. For computational efficiency, we employ a leaky integrate-and-fire (LIF) neuron model, though the NEF can be applied for arbitrary neuron models. Equation (4) is referred to as an encoding equation because it describes how a vector space, in this case x, is encoded into neural spikes. The NEF assumes a least-squares optimal linear decoding to reconstruct x or any nonlinear function thereof, f(x). Thus, we must find the decoders d_i^f , such that

$$E = \frac{1}{2}\int [f(x) - \sum_i a_i(x) d_i^f]^2 dx \tag{5}$$

is minimized. Finding the decoders in this manner then provides us with a way to estimate any vector f(x) given the activities from the encoding equation. We can write this as the decoding equation:

$$\widehat{f(x)} = \sum_i a_i(x) d_i^f \tag{6}$$

where N is the number of neurons in the group and $\widehat{f(x)}$ is the estimate of f(x) where x is the input driving the neurons. Recall that our purpose in defining the representation of a vector space in a neural population is to use it to compute a function between two populations. If we define the encoding and decoding for groups A and B using equations (4) and (6), we can substitute the decoding of A into the encoding of B, thereby deriving connection weights. In addition, if the function we wish to compute is linear, we can include the relevant linear operator in the weight equation. The weight equation for computing any combination of linear and nonlinear functions is then:

$$\omega_{ij} = d_i^f \alpha_j L e_j \tag{7}$$

where i indexes the neurons in group A and j indexes the neurons in B, f is any nonlinear function and L is any D_B x D_A linear operator, where D_A and D_B are the dimensionalities of the two vector spaces.

It is worth noting that these representations and computations can be implemented to any desired precision, by adding enough neurons. Specifically, the root mean-squared-error goes down as 1/N (Eliasmith & Anderson, 2003). One of the main concerns of this paper is to demonstrate that the operations required for representing human-scale lexical structure can be done with a reasonable number of neurons.

It is straightforward to use the NEF to create networks of spiking neurons for computing the inverse and circular convolution operations (Eliasmith, 2005). However, neurally implementing an associative memory requires a specific application of these methods, which we will outline in the next section.

Neural Associative Memory

There are several ways in which associative memories can be implemented (see (Lansner, 2009) for a review). Recently, (Stewart et al., 2010) used the NEF to construct an efficient, fast autoassociative (a.k.a. cleanup) memory out of spiking neurons, and this approach can be trivially extended to construct an associative memory. Moreover, they demonstrate that this approach significantly outperforms a linear associator, a direct function approximator and a standard multi-layer perceptron. However, that paper only considers lexicons up to 10,000 items, and does not discuss any actual lexical processing, as is our focus here.

Given a noisy version of an ID-vector as input, we want our associative memory to output a clean version of the corresponding semantic pointer. A simple algorithm that achieves this is to take the dot product of the input vector with each of the ID-vectors in the vocabulary, threshold these values (set to 0 all values below some fixed threshold), multiply each semantic pointer vector by its corresponding thresholded value, and add all the resultant vectors together to obtain a single D-dimensional vector. If the input vector is only similar to one of the ID-vectors, then all of the dot products will be thresholded except for one and the output vector will be equal to the correct semantic pointer.

We can use the NEF to implement this algorithm in spiking neurons as follows. Assign each synset a small ($\sim$20) population of neurons. Then we set the preferred direction vector of each neuron equal to the ID-vector for the synset that the neuron is assigned to. Equation (4) shows the activities of each population can be seen as encoding the similarity between the input vector and the population's assigned ID-vector. To determine the weight matrices between these populations and the output population, we first minimize equation (5) with f set to a thresholding function to find optimal decoders, and then substitute these into equation (7) with L set to semantic pointer of the population's assigned synset. Thus, the output of a population with ID-vector e and semantic pointer s is a neural reconstruction of $threshold(x^T e) \cdot s$. Summing the output of all the association populations is implicitly performed by the dendrites of the neurons in the output population.

The Model

The core of the model is a network of spiking neurons, constructed using the techniques outlined above, which, given a semantic pointer corresponding to a WordNet synset and a query vector corresponding to a relation type, returns the semantic pointer corresponding to the target of the relation. This network can be used to traverse the WordNet hierarchy by running it recursively, with the output of the last run used as input on the next run. The tasks of moving the output into the input, controlling which relation goes into the query vector population, etc, are not investigated here as they are peripheral to our central concern of representing human-scale structured knowledge in a biologically plausible manner.

A schematic diagram of the model is depicted in Fig-

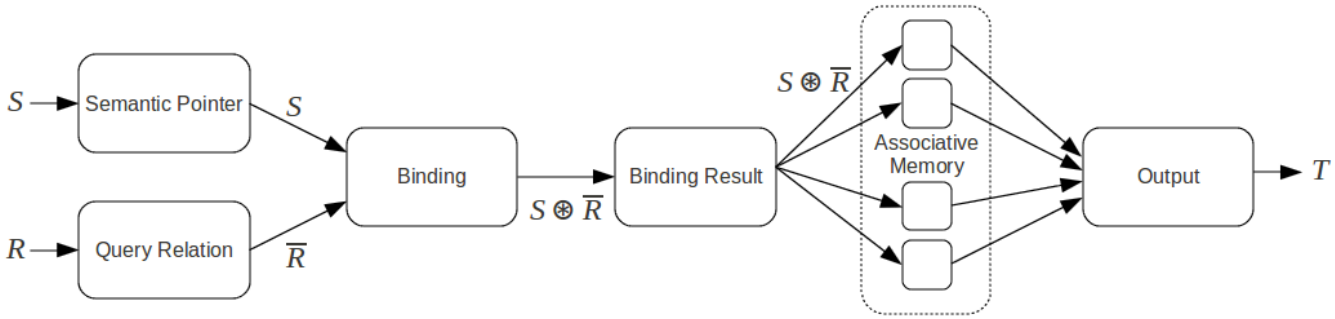

Figure 1: The network of spiking neurons that traverses the WordNet graph. Assume $S = R \circledast T_{ID} + U \circledast V_{ID}$ where R, T_{ID}, U, and V_{ID} are all ID-vectors and T is the semantic pointer corresponding to T_{ID}. All nodes represent neural populations.

ure 1. The nodes correspond to populations of spiking neurons which represent and manipulate 512-dimensional vectors. All neurons employ the leaky integrate-and-fire neuron model. Each time the model was run to perform a single edge traversal, it was simulated for 100 ms with a simulation timestep of 1 ms, after which the vector represented by the Output population was taken to be the output of the model. The Binding node, which performs a circular convolution between two vectors, contains 51,400 neurons, and each of the other 4 nodes outside the associative memory contain 25,600 neurons. The associative memory contains 117,659 populations, one for each WordNet synset, with 20 neurons each, resulting in a grand total of 2,506,980 neurons. This is equivalent to approximately 14.7 mm^2 of cortex, much smaller than previous neural approaches to structured representation, which required on the order of 500 cm^2 cortex (as there are about 170,000 neurons per mm^2; (Eliasmith, in press)).

Experimental Results

We performed three experiments on the model to test different aspects of its performance. For each experiment, a *trial* consists of using the network to answer a single question about the WordNet graph (the question is different for each experiment). A *run* consists of a group of trials. For each experiment we perform some number of runs, calculate the performance on each run as the percentage of trials on which the model answered correctly, and report the mean performance over all the runs. We employ a bootstrapping method to obtain 95% confidence intervals on the mean performance. This data can be seen in Table 1. The model was perfectly successful on Experiment 1, and nearly so on Experiments 2 and 3.

Experiment 1: Decoding Accuracy

This test investigates how many of the 117,659 concepts in the WordNet can be accurately decoded. For this experiment, we present the model with a semantic pointer corresponding to a randomly chosen synset and an ID-vector corresponding to a relation type, and see if the model returns the semantic pointer corresponding to the target of that relation. For example, we might present the network with the semantic pointer for **dog** and the ID-vector for the relation **isA** and see if the network returns the semantic pointer for **canine**. To be considered correct, the returned vector must have a larger dot product with the correct semantic pointer than with any incorrect semantic pointer in the vocabulary, and this dot product must pass a threshold of 0.7. We ran 20 runs, each of which consisted of 100 trials, amounting to 2000 edge traversals.

Experiment 2: Hierarchy Traversal

This experiment is designed to test the model's ability to traverse the network to arbitrary depth. To that end, we use the model to answer the following question: given two synsets and a relation type, can the second synset be reached from the first synset solely by following links of the specified type? We present the model with the semantic pointer corresponding to the first synset as well as the ID-vector for the given relation type. Then we run the model, and compare the output vector to the semantic pointer for the second synset. If they are the same (their normalized dot product is above a fixed threshold), then the model responds with a Yes. If not, we feed the output vector back into the model as the new semantic pointer and run the model again. This process is repeated until the model returns a vector with a norm below a fixed threshold. If the model reaches this point, it responds with a No. Here it is especially important that the decoded semantic pointer be very similar to the correct semantic pointer since we recursively use the output, and large errors would build up with successive edge traversals. Our tests were performed using only the **isA** relation type as it is the most prominent in WordNet and permits the deepest traversals. We ran 20 runs, each consisting of 20 positive examples (the second synset could be reached in the actual WordNet graph), and 20 negative examples.

Table 1: Experimental Results

Experiment	% correct	95% CI	
		lower	upper
1. Decoding Accuracy	100.0	100.0	100.0
2. Hierarchy Traversal	95.5	94.3	96.9
3. Sentence Encoding	99.6	99.3	99.8

Experiment 3: Sentence Encoding

The final experiment we performed was designed to confirm that this method of knowledge representation is flexible enough to allow concepts to bind to arbitrary roles while still encoding the thousands of relationships between themselves. To this end, we test whether the network can accurately decode a crude approximation of a sentence, consisting of synsets bound to one of six different roles. To build a sentence, we randomly choose roles for inclusion, each with a different probability, and then synsets are randomly chosen to fill the selected roles. Each role type is assigned an ID-vector, in the same way that ID-vectors are assigned to relation types. A semantic pointer for the sentence is created by binding synset ID-vectors to role ID-vectors in the usual way. We then present the network with the semantic pointer for the sentence and the ID-vector for a role, and see if the vector it outputs is the same as the semantic pointer for the concept filling that role in that sentence. To determine the correctness of a particular decoding, we used the same criteria as in Experiment 1. We ran 20 runs, each of which consisted of constructing 30 sentences and asking the model about each role therein, amounting to 2411 edge traversals.

Conclusion

These empirical results demonstrate that our spiking neural network can accurately represent structured knowledge representations approaching the scale of those found in an adult human. Moreover, our's is the only approach with neural resource requirements that fall within the range of biological plausibility.

Acknowledgments

Funding for this work was provided by the National Science and Engineering Research Council of Canada, Canada Research Chairs, the Canadian Foundation for Innovation and the Ontario Innovation Trust.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* Oxford University Press.

Doumas, L. A. A., Hummel, J. E., & Sandhofer, C. M. (2008). A theory of the discovery and predication of relational concepts. *Psychological Review*, *115*, 1–43.

Eliasmith, C. (2005). Cognition with neurons: A large-scale, biologically realistic model of the Wason task. In G. Bara, L. Barsalou, & M. Bucciarelli (Eds.), *Proceedings of the 27th annual meeting of the cognitive science society* (pp. 624–630).

Eliasmith, C. (in press). *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation and dynamics in neurobiological systems*. Cambridge, MA: MIT Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science*, *338*(6111), 1202-1205.

Fellbaum, C. (1998). *Wordnet: an electronic lexical database*. Cambridge, Massachusetts: MIT Press.

Gayler, R. W. (2003). Vector Symbolic Architectures answer Jackendoff's challenges for cognitive neuroscience. In P. Slezak (Ed.), *Iccs/ascs international conference on cognitive science* (pp. 133–138).

Hinton, G. (2010). Where do features come from? In *Outstanding questions in cognitive science: A symposium honoring ten years of the david e. rumelhart prize in cognitive science.* Cognitive Science Society.

Jackendoff, R. (2002). *Foundations of language: Brain, meaning, grammar, evolution*. Oxford University Press.

Lansner, A. (2009, March). Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations. *Trends in neurosciences*, *32*(3), 178–86.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain* (1st ed.). The MIT Press. Paperback.

Plate, T. A. (2003). *Holographic reduced representations*. Stanford, CA: CSLI Publication.

Shastri, L., & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings. *Behavioral and Brain Sciences*, *16*, 417–494.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, *46*, 159–217.

Stewart, T., & Eliasmith, C. (2011). Compositionality and biologically plausible models. In W. Hinzen, E. Machery, & M. Werning (Eds.), *Oxford handbook of compositionality.* Oxford University Press.

Stewart, T., Tang, Y., & Eliasmith, C. (2010). A biologically realistic cleanup memory: Autoassociation in spiking neurons. *Cognitive Systems Research*.

van Der Velde, F., & de Kamps, M. (2006). Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, *29*(29), 37–108.

von der Malsburg, C. (1981). *The Correlation Theory of Brain Function* (Vol. Internal R; Tech. Rep.). Abteilung fiir Neurobiologie, MPI for Biophysical Chemistry.