A Neurally Plausible Encoding of Word Order Information into a Semantic Vector Space

Peter Blouw (pblouw@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Center for Theoretical Neuroscience, University of Waterloo
Waterloo, ON N2L3G1 Canada

Abstract

Distributed models of lexical semantics increasingly incorporate information about word order. One influential method for encoding this information into high-dimensional spaces uses convolution to bind together vectors to form representations of numerous *n*-grams that a target word is a part of. The computational complexity of this method has led to the development of an alternative that uses random permutation to perform order-sensitive vector combinations. We describe a simplified form of order encoding with convolution that yields comparable performance to earlier models, and we discuss considerations of neural implementation that favor the use of the proposed encoding. We conclude that this new encoding method is a more neurally plausible alternative than its predecessors.

Keywords: semantic memory; convolution; random permutation; vector space models; distributional semantics

Introduction

The well-known 'semantic space' approach to modeling word meanings is frequently employed by researchers interested in understanding how the brain represents lexical information. At its most simple, the approach involves encoding word co-occurrence statistics from natural language corpora into a set of high dimensional vectors (e.g. Landauer & Dumais, 1997; Lund & Burgess, 1996; Jones & Mewhort, 2007). The spatial relationships between such vectors are then taken to reflect semantic relationships amongst corresponding words. Experiments involving semantic space models have produced impressive results matching human data from studies of category typicality (e.g., Jones & Mewhort, 2007) and synonym identification (e.g., Landauer & Dumais, 1997), amongst other things.

However, one concern with the traditional semantic space approach is that it fails to take into account information about how words are sequentially related to one another (Jones & Mewhort, 2007). For example, the latent semantic analysis (LSA) model developed by Landauer and Dumais (1997) functions by building a word-document frequency matrix that treats all words occurring in a single document equivalently. Similarly, Lund and Burgess' (1996) hyperspace analog to language (HAL) model simply counts the frequency of words occurring within a multi-word window around a target term. This indifference to sentence structure has led to HAL and LSA being referred to as 'bag of words' models of lexical semantics (Jones & Mewhort, 2007; Recchia et al., 2010).

More recently, two techniques have been developed to incorporate word order information into semantic vectors. The first, developed by Jones and Mewhort (2007), uses circular convolution (proposed by Plate (2003) as a vector binding operation) to create vector representations of the numerous *n*-grams a target word is a part of. The second, developed by Sahlgren, Holst, and Kanerva (2008), uses random vector permutation to index the positions of neighboring words in relation to a target word. Functionally, the two approaches are quite similar, but random permutation is much more computationally efficient than convolution (Sahlgren, Holst, & Kanerva, 2008). Moreover, a recent analysis indicates that convolution and random permutation offer similar degrees of accuracy during information retrieval, and that they perform comparably on a set of basic semantic tasks involving synonym identification (Recchia et al., 2010).

Given that computational efficiency favors the use of random permutation, the aim of this paper is to develop a simplified version of convolution encoding that can replicate many of the important functional properties of Jones and Mewhort's (2007) method. More specifically, we use convolution with position-indexing vectors to produce a single *n*-gram for each occurrence of a target word in a corpus (cf. Sahlgren, Holst, & Kanerva, 2008). Encoding a single *n*-gram per word occurrence is much simpler than Jones and Mewhort's technique of encoding multiple *n*-grams per word occurrence, and we demonstrate that this simplification provides good model performance on a range of order-specific tasks involving phrase-completion.

In addition, we argue that our encoding is more biologically plausible for two reasons:

1) All of the required vector representations can be instantiated using simulated spiking neurons.

2) All of the required computations on these representations can also be instantiated using simulated spiking neurons.

To substantiate these claims, we rely on prior work. Eliasmith and Anderson (2003) describe a method for representing and transforming high dimensional real-valued vectors in neural systems through a combination of the non-linear encoding of a signal into a pattern of neural spikes, and the weighted linear decoding of these spikes. Simple operations such as vector addition are easily implemented using these methods, and Eliasmith (2005) extends such work to describe a neural implementation of the circular convolution operation. Since our encoding method utilizes only circular convolution and vector addition, these remarks indicate that it is therefore a neurally plausible method.

In contrast, the approach of Sahlgren, Holst and Kanerva employs binary vectors, which are not naturally implemented in neural models (Stewart & Eliasmith, 2012). Moreover, the approach of Jones and Mewhort employs a series of computations that are arguably too complex to scale appropriately if implemented in neurons. Our position encoding approach, on the other hand, has been utilized in a portion of what is currently the world's largest functional brain model (Eliasmith, et al., 2012), capable of a range diverse tasks involving perception, cognition, and action.

In what follows, we first review the convolution-based encoding algorithm presented by Jones and Mewhort (2007), along with the random permutation algorithm presented by Salhgren, Holst, and Kanerva (2008). We then introduce our own encoding algorithm. Next, we report results from a series of simulations conducted to assess model performance. We conclude that convolution with position indices offers an equally useful but more biologically plausible strategy for incorporating order information into semantic space models.

Two Approaches to Encoding Word Order

The main challenge facing efforts to encode syntactic information into high-dimensional spaces is to find an appropriate, order-preserving mathematical operation for recursively combining vectors. Given that standard vector operations, such as superposition, are inadequate for this purpose, researchers have proposed a number of multiplicative binding methods instead. Examples include Smolensky's (1990) tensor products, Kanerva's (1994) binary spatter codes, and Plate's (2003) holographic reduced representations. Plate's approach has been particularly attractive to researchers interested in language because of its use of circular convolution, which ensures that all recursively bound vectors are of the same dimensionality. In absence of preserved dimensionality, it becomes difficult to compare vectors representing differently structured linguistic objects (e.g. phrases of different lengths; Jones & Mewhort, 2007).

Before getting into the details of encoding with convolution and random permutation, it is worth noting that the point of departure for comparing the two methods is Jones and Mewhort's (2007) BEAGLE[1] model, which assigns each word in a modeled corpus a unique environmental vector (e), along with a zero-valued memory vector (m). Each time a word is encountered in the corpus, its memory vector is updated with context information provided through the superposition of the environmental vectors for every other word in the surrounding sentence. Simultaneously, the memory vector is also updated with a vector describing the ordering of the target word in relation to a limited range neighbors. As whole, the process conforms to the following expression:

$$m_i = m_i + c_i + o_i \qquad (1)$$

where i indexes the word being represented, while c_i and o_i refer to vectors describing context and order information for a given word occurrence.[2] The primary difference, then, between the approaches of Jones & Mewhort (2007) and Sahlgren, Holst, and Kanerva (2008), is in the calculation of o_i. In BEAGLE, o_i incorporates a range of n-grams that a target word is a part of. To give an example of how this works, consider the sentence 'make hay while the sun shines' and the target word 'hay'. The order vector, o_{hay}, is then calculated as the sum of various n-grams that 'hay' is a part of:

$$\begin{aligned} bigram_1 &= e_{make} * \Phi \\ bigram_2 &= \Phi * e_{while} \\ trigram_1 &= e_{make} * \Phi * e_{while} \\ trigram_2 &= \Phi * e_{while} * e_{the} \\ ngram_i &= \ldots \end{aligned}$$

where, * denotes the circular convolution operation, Φ denotes a placeholder vector for the target word, and n sets size of the window around the target word from which order information is drawn. The value of n is typically set to 7.

Overall, this method is quite computationally expensive given that each word occurrence prompts the generation of numerous sequences of convolutions, each of which must be computed in $O(n \log n)$ time (Jones & Mewhort, 2007). Moreover, because convolution is a commutative operation, permutations are applied to distinguish vectors of the form A * B and B * A. This adds an additionally layer of complexity when encoding large sequences of ordered vectors.

In light of this computational complexity, Sahlgren, Holst, and Kanerva's (2008) proposal is to recursively apply a random permutation to the environmental vectors to indicate their position relative to the target word. The random permutation, Π, scrambles the order of the elements in a vector, and its recursive application indexes positions at varying distances from the target word:

$$o_{hay} = \Pi^{-1} e_{make} + 0 + \Pi^{1} e_{while} \ldots + \Pi^{4} e_{shines}$$

Here, the positive superscripts indicate the number times the permutation is applied to an environmental vector, and the negative superscripts indicate the number of times the inverse of the permutation is applied. One important feature of this method is that each occurrence of a target word in the

[1] The acronym stands for 'bound encoding of the aggregate language environment'.

[2] The context and order vectors are normalized prior to being combined and incorporated into the memory vector.

corpus results in the memory vector being updated with only a single *n*-gram containing every word in the order window. The resulting order vector, *o*, is thus structurally quite different from vectors produced through the summing of multiple *n*-grams (Sahlgren, Holst, & Kanerva, 2008).

For information retrieval in this framework, the inverse of a particular position permutation is applied to a memory vector. This process yields a vector that is most similar to environmental vectors that have been frequently bound into the memory vector in this position. Thus, one can extract information about which words are likely to occur in various positions around a target word. For example, $\prod^{-1} m_{hay}$ would yield a vector most similar to words that have frequently been bound into the first position succeeding 'hay' in various order vectors generated over the course of scanning the corpus. Depending on the statistical properties of this corpus, a comparison (i.e. cosine measure) between $\prod^{-1} m_{hay}$ and the environmental vectors will likely yield an environmental vector such as e_{bale} as most similar.

Overall, when comparing these methods for generating memory vectors, three things are important to keep in mind. First, there are a number of further differences between BEAGLE and Sahlgren, Holst, and Kanerva's model beyond the use of random permutation for order encoding. For example, the latter model uses binary environmental vectors, while Jones and Mewhort's model uses environmental vectors whose elements are picked from a Gaussian distribution of a mean of zero and variance equal to 1/D.[3] Moreover, Sahlgren, Holst, and Kanerva apply a smaller window for calculating context information that ignores sentence boundaries. These differences limit the ability to conduct performance comparisons based on the use of random permutation alone.

Second, to the extent that such comparisons have been made, they focus almost exclusively on storage capacity measures and performance on simple synonym identification tasks. However, one of the more compelling attributes of the BEAGLE model is its ability to reflect experimental effects involving things like category typicality, priming, and semantic constraints on stem completion. It has not been demonstrated that models built using random permutation have comparable capabilities.

Third, the BEAGLE model is computationally expensive, but uses real-valued vectors (which are efficiently implementable in a biologically plausible network; Eliasmith & Anderson, 2003), whereas the permutation model is computationally efficient, but uses binary vectors (which have not been demonstrated to be efficient to implement biologically). Past work has not proposed a representation that is both computationally and biologically efficient.

Here, we describe a new representation that is comparable to the BEAGLE model in that it preserves the functional properties of its memory vectors, but it uses a single *n*-gram order encoding method that is structurally similar to Sahlgren, Holst, and Kanerva's technique while employing real-valued vectors.

Convolution with Position Vectors

Our proposal is to encode order information with a set of reusable, real-valued, unitary, randomly generated 'position vectors'.[4] These vectors are convolved with environmental vectors and summed to give an order vector of the following form:

$$o_i = \ldots p_{-1} * e_{-1} + 0 + p_1 * e_1 + p_2 * e_2 \ldots \qquad (2)$$

where p_1 is the vector that indexes the first position succeeding the target word, p_{-1} is the vector that indexes the first position preceding the target word, and so forth. e_1, e_2, etc. are the environmental vectors of the words in each position around the target word. Structurally, this approach shares the property of position indexing with the model of Sahlgren, Holst and Kanerva (2008), but computationally, it shares the use of convolution of real-valued vectors with the model of Jones and Mewhort (2007).

To make the proposal clearer, consider again the word 'hay' in the sentence 'make hay while the sun shines'. The order vector produced with our method would be

$$o_{hay} = p_{-1} * e_{make} + 0 + p_1 * e_{while} \ldots + p_2 * e_{shines}$$

Once this order vector is incorporated into the memory vector for 'hay', this memory vector will become slightly more similar to other vectors with have had 'hay' bound into the first position to the right too.

To retrieve order information from a memory vector, we can use one of two methods, both adapted from Jones & Mewhort (2007). The first is to convolve the inverse of a position vector with a memory vector to extract a representation that is most similar to the environmental vectors that have been frequently bound into the memory vector in this position. For example:

$$m_{hay} * p_1^{-1} \approx e_{while}$$

Note that this method can be used to extract words commonly found in any of the twelve positions for which order information is encoded.

The second form of information retrieval involves constructing a probe corresponding to particular ordering around a target word, and then identifying which memory vectors have most frequently encoded the ordering of

[3] These properties are needed to ensure that convolution can be used effectively as an operation for binding and unbinding vectors (Plate, 2003).

[4] To index position, a single unitary vector could also be self-convolved multiple times. This would avoid the use of random vectors for each position, but it is functionally equivalent to the present formulation.

interest. To give an example, one could construct the following probe vector:

$$probe = p_{-1} * e_{make} + 0 + p_1 * e_{while} + ... + p_3 * e_{shines}$$

If this vector is compared to all memory vectors generated from the corpus, it will match most closely with words that have frequently encoded the order sequence 'make ___ while the sun shines'. Provided that the corpus does not contain a multitude of words that repeatedly occupy the blank position in relation to the same the surrounding words, the comparison will return the memory vector m_{hay} as the closest match.

Overall, information retrieval is made quite simple when position encoding is conducted via convolution with position vectors. As important, however, is whether or not the encoding enables good model performance.

Simulations

We test the effects of the position encoding method for performance on a range of tasks involving semantic similarity and phrase completion. As per Jones and Mewhort (2007), context vectors are calculated as the superposition of environmental vectors in the sentence surrounding a target word, and environmental vectors are randomly generated with elements drawn from a Gaussian distribution. A list of stop words is used to prevent frequently occurring function words from being overrepresented in the context vectors, and order information is calculated using position indices ranging from -6 to +6. This range is chosen because it captures the same set of words that would be included in order vectors calculated using Jones and Mewhort's original method. Finally, context vectors and order vectors are normalized prior to inclusion in the overall memory vector for a given word.

All simulations are run, for efficiency, on a subset of the same TASA corpus used in tests of both BEAGLE and Sahlgren, Holst, and Kanerva's (2008) random permutation model. Approximately 27,000 unique words are modeled using roughly 110,000 sentences, and words occurring less than twice in the corpus are ignored to exclude misspellings and typographical errors.

A Nearest Neighbors Task

As an initial qualitative assessment of model performance, we calculated the nearest neighbors to the memory vectors for four common words found in the TASA corpus. We chose the same four words used in Table 3 of Jones and Mewhort (2007). The results, shown in Table 1 below, indicate that encoding order information with position vectors instead of an array of *n*-grams results in plausible model performance for each of the four words. All reported activation values are cosines of the angle between two vectors in the semantic space. The context space is comprised of memory vectors only updated with context information, while the order space is comprised of memory vectors only updated with order information. The combined space includes memory vectors calculated in accordance with Equation 1.

As with the comparison between BEAGLE and the model of Sahlgren, Holst, and Kanerva (2008), subtle differences in things like the selection of stopwords and the formation of the environmental vectors make quantitative comparisons impractical, so we present these results as an independent demonstration of model performance.

Table 1: Nearest Neighbors in Three Spaces

Context		Order		Combined	
EAT					
food	0.69	get	0.89	get	0.78
get	0.65	buy	0.87	make	0.75
animals	0.63	make	0.86	take	0.70
need	0.62	keep	0.86	keep	0.69
make	0.61	meet	0.85	find	0.69
CAR					
came	0.65	nation	0.89	house	0.75
back	0.64	village	0.88	road	0.73
road	0.64	fire	0.88	big	0.73
one	0.63	family	0.88	little	0.71
way	0.63	story	0.88	dog	0.70
READING					
read	0.66	writing	0.72	writing	0.68
book	0.61	making	0.67	that	0.61
writing	0.61	business	0.64	your	0.61
skimming	0.59	power	0.62	or	0.61
may	0.56	food	0.62	this	0.59
SLOWLY					
little	0.63	quickly	0.75	quickly	0.62
around	0.63	again	0.67	and	0.60
back	0.62	ran	0.65	down	0.60
across	0.60	to	0.65	then	0.59
move	0.59	brought	0.65	to	0.59

Retrieval with Decoding

Retrieval through decoding, again, involves convolving a memory vector with the inverse of a position vector, and then comparing the output of this process to a library of environmental vectors to find closest matches. In this simulation, we use the decoding retrieval method to find the most likely word to occur both before and after a particular target word. Results are reported in Table 2.

One point to note about these decoding results is that the activation values for the words in each column indicate non-random correspondence with the target word if the similarity value is greater than ~0.1 (see Jones and Mewhort, 2007, p. 13). Accordingly, the decoding does a good job of picking out words that are likely to follow before or after a given word.

Table 2: Decoding Around a Target Word

Word Before		Word After	
LUTHER			
martin	0.29	king	0.21
straightening	0.17	gravity	0.17
latest	0.17	1733	0.16
coinage	0.16	puff	0.16
so-called	0.16	conscience	0.16
KING			
the	0.54	was	0.19
experienced	0.17	tens	0.17
boundaries	0.17	bowing	0.17
kites	0.17	lawfully	0.17
donor	0.16	pasture	0.16

Retrieval with Resonance

Resonance retrieval, again, involves constructing a probe by superposing a number of bound environmental and position vectors. This probe vector is then compared to all of the memory vectors to find items that have frequently occurred within the sequence of words described by the probe.

Table 3: Resonance Around a Target Word

Word Before		Word After	
KING			
rex	0.38	midas	0.42
luther	0.22	tut	0.42
rumbles	0.17	aietes	0.39
hamlet	0.17	farouk	0.36
oyster	0.16	richards	0.31
PRESIDENT			
vice	0.32	eisenhower	0.45
activist	0.20	lincoln	0.31
egypts	0.19	coolidge	0.27
middle-of-the-road	0.19	johnson	0.25
dove	0.18	nixon	0.23
WAR			
spanish-american	0.31	II	0.49
civil	0.29	bonnet	0.21
post-world	0.27	hysteria	0.19
pre-civil	0.26	whoops	0.19
post-civil	0.23	1898	0.18
SEA			
caspian	0.22	anemone	0.38
Aegean	0.22	level	0.27
mediterranean	0.19	gull	0.26
foaming	0.17	anenomes	0.24
sensitivity	0.16	captains	0.24

To assess model performance with resonance, we simulate a task involving retrieval around a set of four target words drawn from Table 4 of Jones and Mewhort (2007). The results from this simulation are presented in Table 3. Despite the intrusion of a few unexpected items into these lists of nearest matches (e.g. 'sensitivity'), the overall trend here provides further evidence that order encoding with position vectors can produce a functioning semantic space model.

Phrase Completion with Resonance

To go beyond the retrieval of words either immediately to the left or to the right of a target word, we next simulate a set of tasks in which probe vectors corresponding to short phrases are compared to the memory vectors. Initially, only a limited amount of information is included in the probe vector, but subsequently, the probe is enriched to represent a more and more specific order sequence (see Jones & Mewhort, 2007). As more information is incorporated into the probe in this way, the model increasingly converges on a single word that best fits the blank region in the probe phrase. We use phrase materials drawn from Jones and Mewhort (2007). Results are reported in Table 4 below.

Once again, the model generally meets performance expectations. Preliminary results also indicate that the model generally performs well with other phrases similar to the ones shown. Further work is ongoing in this area.

Discussion

At this point, it seems clear that the method of encoding with position vectors performs well enough to be considered a plausible alternative to earlier methods. However, it is worth considering the criteria by which one might select amongst the three forms of encoding discussed in this paper. Computational efficiency, again, favors the use of a single *n*-gram encoding method like random permutation or encoding with position vectors.

Then, to decide between convolution and random permutation, one could look to performance measures of the sort just examined. Here, position vector encoding has the advantage of a demonstrated ability to perform a variety of phrase completion tasks; the performance credentials of random permutation have yet to be comparably established. It is possible that random permutation supports the same degree of functionality as demonstrated here, and future work might bear out such a prediction.

However, even if this is the case, we think that independent considerations of neural implementation favor the use of the position vector encoding method. First, note that vector space models of language have appealed to cognitive researchers in part because they possess certain properties suggestive of neural plausibility (Jones & Mewhort, 2007; Recchia et al., 2010). Connectionist models, for example, have long been used to implement computations defined over vectors, and one of the main attractions of these models is their use of neurally inspired processing mechanisms. So, because semantic space models

Table 4: Highest Word Activations as an Order Sequence is Filled in Around a Target Position

Phrase	Activations					
emperor [penguins]	yuan	0.26	penguins	0.26	caligula	0.20
[penguins] have	planaria	0.34	threepio	0.27	astronomers	0.26
the emperor [penguins] have come to their breeding grounds	penguins	0.34	yaun	0.31	annelida	0.27
although [ostriches]	gauges	0.21	democratically	0.20	tsumanis	0.18
although [ostriches] cannot	pretends	0.16	raindrops	0.16	democratically	0.16
although [ostriches] cannot fly they have other skills	ostriches	0.18	assent	0.18	caved	0.16

are constructed through computations defined over vectors, and connectionist models can implement such computations, it follows that semantic space models can share to some extent in the claim of being consistent with how the brain processes information.

Second, models with a high degree of neural plausibility have been built using vector symbolic architectures that employ real-valued vectors and convolution as a binding operator. The same cannot be said for binary vectors. For instance, neurally implemented convolution operations play a key role in a recent model of working memory (Choo & Eliasmith, 2010), and more significantly, what is currently the world's largest functional brain model (Eliasmith et al., 2012). So, the argument in favor of using convolution with position vectors to encode word order into semantic space models is straightforward: doing so is consistent with the architectural principles that guide state-of-the-art models of complex cognition. Put simply, there is a good deal evidence from these models that the convolution operation accommodates the computational constraints of neural systems.

Together with the demonstrated functionality of semantic space models built using convolution encoding, we think that these considerations of neural implementation provide a compelling case in favor of the method we demonstrate here. Convolution with position vectors provides an approach to building an order-sensitive semantic vector space that is functional, neurally plausible, and relatively computationally efficient. We leave it to future work to determine whether methods utilizing random permutation can display a similar range of strengths.

Acknowledgments

This research was supported by the Social Sciences and Humanities Research Council of Canada.

References

Choo, F.X., & Eliasmith, C. (2010). A spiking neuron model of serial order recall. *Proceedings of the 32nd Annual Conference of the Cognitive Science Society.* (pp. 2188-2193)

Eliasmith, C. (2005). Cognition with neurons: A large-scale biologically plausible model of the Wason card task. *Proceedings of the 27th Annual Conference of the Cognitive Science Society* (pp. 624-630).

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

Eliasmith, C., Stewart, T., Choo, F.X., Bekolay, T., DeWolf, T., Tang, Y., & Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, 338.6111, 1202-1205.

Jones, M.N. & Mewhort, D. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, *114.1*, 1-37.

Kanerva, P. (1994). The spatter code for encoding concepts at many levels. *Proceedings of the International Conference on Artificial Neural Networks.* (pp. 226-229). Sorrento, Italy: Springer-Verlag.

Landauer, T., and Dumais, S. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review, 104.*2, 211–240.

Lund, K. & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavioral Research Methods,* 28.2, 203-208.

Plate, T.A. (2003). *Holographic reduced representations: distributed representations for cognitive structures*. Stanford, CA: CSLI Publications.

Recchia, G., Jones, M., Sahlgren, M., & Kanerva, P. (2010). Encoding sequential information in vector space models of semantics: Comparing holographic reduced representations and random permutation. *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 865-870).

Sahlgren, M., Holst, A., & Kanerva, P. (2008). Permutations as a means to encode order in word space. *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 1300-1305).

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46, 159-216.

Stewart, T. & Eliasmith, C. (2012). Compositionality and Biologically plausible models. In W. Hinzen, E. Machery, & M. Werning (eds.) *Oxford Handbook of Compositionality* (pp. 596-615). Oxford: Oxford University Press