

# Permutations as a Means to Encode Order in Word Space

Magnus Sahlgren (mange@sics.se)      Anders Holst (aho@sics.se)

Swedish Institute of Computer Science  
Kista, Sweden

Pentti Kanerva (pkanerva@csl.stanford.edu)

Center for the Study of Language and Information  
Stanford, California, USA

## Abstract

We show that sequence information can be encoded into high-dimensional fixed-width vectors using permutations of coordinates. Computational models of language often represent words with high-dimensional semantic vectors compiled from word-use statistics. A word’s semantic vector usually encodes the contexts in which the word appears in a large body of text but ignores word order. However, word order often signals a word’s grammatical role in a sentence and thus tells of the word’s meaning. Jones and Mewhort (2007) show that word order can be included in the semantic vectors using holographic reduced representation and convolution. We show here that the order information can be captured also by permuting of vector coordinates, thus providing a general and computationally light alternative to convolution.

**Keywords:** word-space model; distributional hypothesis; context vector; semantic vector; holographic reduced representation; random indexing; permutation

## Word Space

A popular approach to model meaning similarities between words is to compute their distributional similarity over large text data. The underlying assumption—usually referred to as the *distributional hypothesis*—states that words with similar distribution in language have similar meanings, and that we can quantify their semantic similarity by comparing their distributional profiles. This is normally done by collecting word occurrence frequencies in high-dimensional *context vectors*, so that distributional similarity between words may be expressed in terms of linear algebra. Hence such models are referred to as *word-space models*.

There are a number of well-known models in the literature; the most familiar ones are HAL (Hyperspace Analogue to Language; Lund, Burgess and Atchley, 1995) and LSA (Latent Semantic Analysis; Landauer and Dumais, 1997). These models represent two fundamentally different approaches for producing word spaces, and the difference lies in the way context vectors are produced. HAL produces context vectors by collecting data in a words-by-words matrix that is populated with frequency counts by noting co-occurrences within a sliding *context window* of word tokens (normally 10 tokens wide). The context window is direction sensitive, so that the rows and the columns in the resulting matrix represent co-occurrences to the right and left of each word. Each row-column pair is then concatenated and the least variant elements discarded. LSA, on the other hand, produces context vectors by collecting data in a words-by-documents matrix that is populated by noting the frequency of occurrence of words in documents. The resulting matrix is then transformed

by normalizing the frequency counts, and by reducing the dimensionality of the context vectors by truncated singular value decomposition.

Sahlgren (2006) argues that these two types of word-space models produce different semantic representations. The HAL-type of model produces *paradigmatic* representations in which words that occur with similar *other* words get similar context vectors, while the LSA-type of model produces predominantly *syntagmatic* representations in which words that *co-occur* in documents get similar context vectors. Paradigmatically similar words are semantically related, like synonyms and antonyms (e.g. “dark”–“black” and “dark”–“bright”), while syntagmatically similar words tend to be associatively rather than semantically related (e.g. “dark”–“night”). This paper is only concerned with the former type of paradigmatic model that counts co-occurrences at word level.

A commonly raised criticism from a linguistic perspective is that these models are inherently agnostic to linguistic structure. This is particularly true of LSA, which is only concerned with occurrences of words in documents, and thus is thoroughly indifferent to syntax. HAL, on the other hand, incorporates very basic information about word order by differentiating between co-occurrences with preceding and succeeding words. However, most other paradigmatic (HAL-like) models typically do not incorporate order information, and thus do not differentiate between sentences such as “Angela kissed Patrick” and “Patrick kissed Angela”. There have been studies showing how word-space representations can benefit from certain linguistic refinement, such as morphological normalization (Karlgrén and Sahlgren, 2001), part-of-speech tagging (Widdows, 2003) or dependency parsing (Padó and Lapata, 2007), but attempts at incorporating order information have thus far been few.

This paper is inspired by a method of Jones and Mewhort (2007) for including word-order information in paradigmatic word spaces. We introduce a related method that is both general and computationally simple. It is based on the permutation of vector coordinates, which requires much less computing than the convolution operation used by Jones and Mewhort. Both methods are approximate and rely fundamentally on high-dimensional random vectors. They offer a major advantage over their exact counterparts, in that the size of the vocabulary does not need to be fixed in advance. The methods adapt naturally to increases in vocabulary as new data become available.

We first review Jones and Mewhort’s convolution-based

approach and then introduce our permutation-based alternative. We also provide experimental evidence demonstrating that order information produces refined paradigmatic word spaces as compared to merely using proximity, thus corroborating Jones and Mewhort’s results.

### Summary of Jones and Mewhort’s BEAGLE

In a recent fundamentally important study, Jones and Mewhort (2007) show how word-order information can be included in the context vectors for paradigmatic word space representations. Their BEAGLE (Bound Encoding of the Aggregate Language Environment) model represents words with 2,048-dimensional vectors ( $D = 2,048$ ). BEAGLE reads text one sentence at a time and collects two kinds of information for each word in the sentence: *context information* (what are the other words it occurs with), and *order information* (in what order do they occur). At the end of processing each word in the vocabulary—each unique word that appears in the text—is represented by three *memory vectors*, one for context, one for order, and one combining the two. The memory vectors are also called *semantic vectors*.

The three memory vectors are computed with the aid of  $D$ -dimensional auxiliary vectors called *environmental vectors*. Each word in the vocabulary has its own environmental vector, it is set at the beginning of processing, and it does not change thereafter. The environmental vectors are random vectors; their components are normally distributed i.i.d. random variables with 0 mean and  $1/D$  variance. Each word is also assigned a  $D$ -dimensional memory vector that is initially set to the 0-vectors. The environmental vectors then mediate the transfer of information from the sentences to the memory vectors. To see how, we will use Jones and Mewhort’s example sentence

“a dog bit the mailman”

with “dog” as the *focus* word, and we will use the following notation for the various vectors for “dog”:

- dog (lower case) environmental vector, set once at the start
- [dog] context information from the present sentence
- <dog> order information from the present sentence
- [DOG] memory vector for accumulating context information
- <DOG> memory vector for accumulating order information
- DOG (upper case) memory vector combining the two

### Context Information

The example sentence would yield the following context information about “dog”:

$$[\text{dog}] = a + 0 + \text{bit} + \text{the} + \text{mailman}$$

except that there is a list of “stop words” for excluding very frequent (function) words, so that the context information from this sentence becomes

$$[\text{dog}] = 0 + 0 + \text{bit} + 0 + \text{mailman} = \text{bit} + \text{mailman}$$

It is normalized (to vector length 1 using division) and added to the memory vector [DOG]. Due to the nature of vector addition, [DOG] becomes a little more like bit and mailman. Each time “dog” appears in the text it contributes to [DOG] in this manner.

### Order Information

The order information for “dog” is the sum of all  $n$ -grams (up to a limit on  $n$ ) that include “dog.” The  $n$ -grams are encoded with the aid of a place-holder vector  $\Phi$  and convolution  $*$ , where  $\Phi$  is just another environmental vector (see above) and  $*$  is multiplication in mathematical parlance. The convolution is the hallmark of holographic reduced representation due to Plate (2003).

The example sentence yields the following information on the position of “dog,” namely, the bi-, tri-, tetra-, and penta-grams for “dog”:

- <dog><sub>1</sub> =  $a * \Phi$
- <dog><sub>2</sub> =  $\Phi * \text{bit}$
- <dog><sub>3</sub> =  $a * \Phi * \text{bit}$
- <dog><sub>4</sub> =  $\Phi * \text{bit} * \text{the}$
- <dog><sub>5</sub> =  $a * \Phi * \text{bit} * \text{the}$
- <dog><sub>6</sub> =  $\Phi * \text{bit} * \text{the} * \text{mailman}$
- <dog><sub>7</sub> =  $a * \Phi * \text{bit} * \text{the} * \text{mailman}$

Note that the frequent function words, by being grammar markers, are now included. The vector sum

$$\langle \text{dog} \rangle = \langle \text{dog} \rangle_1 + \langle \text{dog} \rangle_2 + \dots + \langle \text{dog} \rangle_7$$

is normalized and added to the memory vector <DOG>, making it a little more like each of the  $n$ -grams <dog><sub>1</sub>, ..., <dog><sub>7</sub>. As above with [DOG], each time “dog” appears in the text it contributes to <DOG> in this manner.

Due to the nature of the convolution, different  $n$ -grams get different encodings that resemble neither each other nor any of the environmental vectors. For example, <dog><sub>7</sub> uniquely and distinctly encodes the fact that the focus word is immediately preceded by “a,” is immediately followed by “bit,” which is immediately followed by “the,” which is immediately followed by “mailman.”

Finally, the combined memory vector is the sum

$$\text{DOG} = [\text{DOG}] + \langle \text{DOG} \rangle$$

and it is sensitive to both proximity and word order.

### Encoding Order with Permutations

Jones and Mewhort’s method of capturing order information—of encoding  $n$ -grams—is based on two ideas. First, the convolution (i.e., multiplication) of vectors  $a$  and  $b$  produces a vector  $a * b$  that is *dissimilar*—approximately orthogonal—to both  $a$  and  $b$ , so that when an  $n$ -gram is added into the memory vector it acts as random noise relative to all other contributions to the memory vector. This is what allows frequent occurrences of the same environmental vector or the same  $n$ -gram vector to dominate the final memory vector. Second, convolution is *invertible*, allowing further analysis of memory vectors. For example, given the vector <DOG>

(or the vector DOG) we can find out what word or words most commonly follow “dog” in the text: when the inverse operator is applied to <DOG> in the right way, it produces a vector that can be compared to the environmental vectors in search for the best match.

This points to other ways of obtaining similar results, that is, to other kinds of environmental vectors and multiplication operations for them. We have used Random Indexing (Kanerva, Kristofersson and Holst, 2000), which is a form of random projection (Papadimitriou et al., 1998) or random mapping (Kaski, 1999). The environmental vectors are high dimensional, random, sparse, and ternary (a few randomly placed 1s and -1s among many 0s)—we call them *Random Index vectors*. Permutation, or the shuffling of the coordinates, can then be used as the “multiplication” operator; it can be used also with other kinds of environmental vectors including those of BEAGLE. See also Gayler (1998) for “hiding” information with permutation in holographic representation. When the coordinates of an environmental vector are shuffled with a random permutation, the resulting vector is nearly orthogonal to the original one with very high probability. However, the original vector can be recovered with the reverse permutation, meaning that permutation is invertible.

Context (word proximity) information can be encoded in Random Indexing with the very same algorithm as used by Jones and Mewhort (i.e., add [dog] = bit + mailman into [DOG]), but the details of encoding order information differ. For example, the order information for the focus word “dog” in “a dog bit the mailman” can be encoded with

$$\langle \text{dog} \rangle = (\Pi^{-1} a) + 0 + (\Pi \text{ bit}) + (\Pi^2 \text{ the}) + (\Pi^3 \text{ mailman})$$

Here  $\Pi$  is a (random) permutation,  $\Pi^{-1}$  is its inverse, and  $\Pi^n$  means that the vector is permuted  $n$  times.

As with  $\langle \text{dog} \rangle_7$  above,  $\langle \text{dog} \rangle$  here encodes the fact that the focus word is immediately preceded by “a,” is immediately followed by “bit,” which is immediately followed by “the,” which is immediately followed by “mailman.” However, the encoding is not unique in a loose sense of the word. Although no other  $n$ -gram produces exactly the same vector for “dog,” the  $\langle \text{dog} \rangle$ -vectors of “dog bit the mailman,” “a dog bit the mailman,” and “a dog bit a mailman,” for example, are similar due to the nature of vector addition. A major advantage of this method is that  $\langle \text{dog} \rangle$  now represents all seven  $n$ -grams (see  $\langle \text{dog} \rangle_i$  above) at once with very little computation. Akin to the BEAGLE algorithm, this algorithm produces a memory vector <DOG> that can be analyzed further, for example, to find out what word or words most commonly follow “dog” in the text.

### Order-Based Retrieval

When memory vectors encode order information, for example when they are computed from context windows using one permutation  $\Pi$  for the words that follow the focus word, and its inverse  $\Pi^{-1}$  for the words that precede it, the memory vectors can be queried for frequent right and left neighbors: for example, what words frequently follow “dog” based on the

memory vector <DOG>? We will refer to this kind of querying as “retrieval” and it is based on the following idea.

Using + and - to denote the two permutations, we note that whenever “dog bit” occurs in the text, the permuted index vector bit+ is added to <DOG> making it a bit more like bit+. To retrieve bit from <DOG> we must first undo the permutation, so we will compare <DOG>- to all *index* vectors. The best-matching index vectors—the ones with the highest cosine scores—will then indicate words that most often follow “dog” in the text.

We also note that the words following “dog” in the text add dog- into their memory vectors, for example, dog- is added to <BIT>. This gives us a second method of searching for words that often follow “dog,” namely, compare dog- to all *memory* vectors and choose the best-matching ones.

## Experiments

We have tested permutations in a number of simulations. The text is the same ten million word TASA corpus as in Jones and Mewhort’s experiments, but it has first been morphologically normalized so that each word appears in base form. As test condition, we use a synonym test consisting of 80 items from the TOEFL (Test Of English as a Foreign Language) synonym part. This is the same test setting as used by Jones and Mewhort, and in many other experiments on modeling meaning similarities between words. The task in this tests is to find the synonym to a probe word among four choices. Guessing at random gives an average score of 25% correct answers. The model’s choice is the word among the four that is closest to the probe word as measured by the cosine between semantic vectors.

The context of a word, including order information, is taken from a window of a fixed width that slides through the text one word at a time without regard to sentence boundaries. A notation such as 2+2 means that the window spans two words before and two words after the focus word. We use fairly narrow context windows in these experiments, since they have been shown in previous studies to be optimal for capturing paradigmatic information (Redington, Chater and Finch, 1998; Sahlgren, 2006).

Context information is encoded as in BEAGLE: it is the sum of the index vectors for the words surrounding the focus word within the window. Thus “two dark brown dogs bit the mailman” yields the context information

$$[\text{dog}] = \text{dark} + \text{brown} + 0 + \text{bite} + 0$$

for “dog” when a 2+2 window is used and function words are omitted. In the following, we refer to such representations as *context vectors*.

The experiments include two ways of encoding order. In one approach we distinguish merely whether a word occurs before or after the focus word. Then only two permutations are used,  $\Pi^{-1}$  with words before and  $\Pi$  with words after. We refer to this as *direction vectors*—note that this corresponds to the direction-sensitive representations used in HAL. In the other approach to encoding order, the permutations progress

through the powers of  $\Pi$  according to the distance to the focus word as shown for  $\langle \text{dog} \rangle$  in the previous section, thus capturing all order information. We refer to such vectors as *order vectors*.

Since our entire approach of encoding order information is based on a single permutation  $\Pi$  and since the index vectors are random, we can use rotation of a vector by one position for  $\Pi$ . Then  $\Pi^2$  means rotating it by two positions,  $\Pi^{-1}$  means rotating by one position in the opposite direction, and so forth.

Unless stated otherwise, all results are average scores from three different runs using different initializations of the random vectors.

### An Example of Order- and Context-Based Retrieval

We wanted to get a sense of the information encoded in the semantic vectors and used the four search words from Jones and Mewhort’s Table 4 to retrieve words likely to precede and to follow them (order information), and words appearing in similar contexts (context information); see Table 1. The table (which is based on one run) was computed with direction vectors based on 3,000-dimensional ternary index vectors with 30 1s and 30  $-1$ s, with a 2+2 context window, and with a frequency threshold of 15,000. The five words with the highest cosines, and the cosines, are shown for each search word. Many of the retrieved words agree with those of Jones and Mewhort. We should note that a word before and a word after can actually be the second word before or after, as in “King (of) England,” because the table is based on direction vectors rather than order vectors.

Table 1: Retrieval by order and context

| Word before       | Word after       | Context-only       |
|-------------------|------------------|--------------------|
| <b>KING</b>       |                  |                    |
| luther .24        | queen .43        | ruler .35          |
| martin .22        | england .25      | prince .27         |
| become .17        | midas .16        | priest .26         |
| french .14        | france .15       | england .26        |
| dr .13            | jr .14           | name .26           |
| <b>PRESIDENT</b>  |                  |                    |
| vice .69          | roosevelt .22    | presidency .57     |
| become .23        | johnson .20      | agnew .30          |
| elect .20         | nixon .18        | spiro .29          |
| goodway .09       | kennedy .15      | admiral .26        |
| former .09        | lincoln .15      | impiety .26        |
| <b>WAR</b>        |                  |                    |
| world .60         | ii .46           | expo .46           |
| civil .48         | independence .10 | innsbruck .44      |
| during .20        | end .09          | disobedience .43   |
| after .19         | over .08         | ii .42             |
| before .10        | altar .07        | ruggedness .39     |
| <b>SEA</b>        |                  |                    |
| mediterranean .39 | level .53        | trophic .50        |
| above .32         | captain .22      | classificatory .45 |
| red .19           | animal .13       | ground .40         |
| black .17         | floor .12        | above .34          |
| north .14         | gull .11         | optimum .33        |

Table 2: Overlap between word spaces.

|                   | Overlap |     |     |     |       |
|-------------------|---------|-----|-----|-----|-------|
|                   | 1+1     | 2+2 | 3+3 | 4+4 | 10+10 |
| Context/Direction | 48%     | 47% | 47% | 46% | 51%   |
| Context/Order     | 48%     | 37% | 32% | 29% | 19%   |
| Direction/Order   | 100%    | 60% | 52% | 49% | 35%   |

### The Overlap Between Context, Direction and Order

In a first set of experiments, we compute the overlap between word spaces produced with context vectors, direction vectors (i.e. using one permutation for all words before the focus word and another for all words after it), and order vectors (i.e. using permutations that progress through the powers of  $\Pi$  according to the distance to the focus word). The overlap is computed as described in Sahlgren (2006): we select 1,000 words from the data at random and for each word we extract the ten nearest neighbors from the context space, the ten nearest neighbors from the direction space, and the ten nearest neighbors from the order space. We then compare the ten-word lists pairwise, count the words they have in common, and average over the 1,000.

Table 2 summarizes the comparison between a context space, a direction space and an order space built using different context windows. As a comparison, the overlap for word spaces produced with context vectors collected with different window sizes are somewhere around 20% (for 1+1 windows vs. 2+2 windows) to 30–40% (for 2+2 windows vs. 3+3 windows; the overlap between paradigmatic and syntagmatic word spaces is much lower—somewhere around 1–10% depending on context size and data). As can be seen in the table, order vectors become increasingly dissimilar to both direction and context vectors as the window size increases, which is an effect of using different permutations for each position in the context window. The overlap between context and direction vectors remains fairly stable around 46–51%. Notice that the overlap between the order and the direction space for a 1+1-sized context window is 100% because the order and direction vectors are identical—one permutation before and another after.

### The Effect of Frequency Thresholding

In Jones and Mewhort’s experiment, function words are excluded from the windows when computing context vectors, but *not* when computing order vectors. We believe that our method of encoding order should benefit from removal of very frequent words, because they dominate our context windows (our  $n$ -grams are encoded with addition). In this experiment, we investigate the effect of frequency thresholding (which is equivalent to filtering function words), for order vectors, direction vectors, context vectors and combined (context + direction) vectors. All vectors are 3,000-dimensional, the index vectors have two 1s and two  $-1$ s placed randomly among the vectors elements, the context

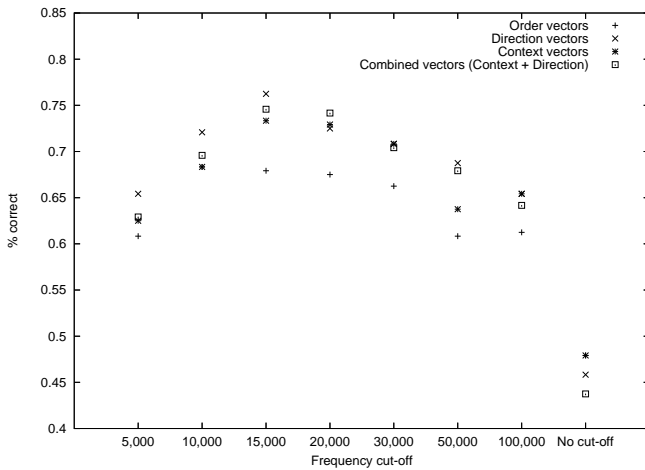


Figure 1: Percent correct answers for different frequency cut-offs.

window is 2+2, and the criterion is the score with TOEFL synonyms. Figure 1 summarizes the results.

The effect of frequency thresholding is apparent for *all* vectors, including direction and order vectors. The results improve drastically when high-frequency words are removed from the context windows by frequency thresholding. We also tested two different stoplists (the SMART information retrieval stoplist containing 571 words,<sup>1</sup> and an enlarged version encompassing 706 words), but they did not improve the performance—in fact, they consistently lead to inferior results compared to using a frequency cut-off at 15,000 occurrences, which removes 87 word types and reduces the vocabulary from 74,187 to 74,100 words. Furthermore, we tested whether removing words with very low frequency had any effect on the results, but consistent with previous research (Sahlgren, 2006), we failed to see any effect whatsoever. Informed by these results, we use a frequency cut-off of 15,000 occurrences in the following experiments.

### The Effect of Dimensionality

Varying the dimensionality of the vectors from 1,000 to 50,000 has a similar impact on all semantic vectors—the results increase with the dimensionality, as can be seen in Figure 2. The best result in our experiments is 80% correct answers on the TOEFL synonyms using direction vectors with a dimensionality of 30,000 (in every case the index vectors have two 1s and two -1s and the context window is 2+2). Note that the direction vectors consistently produce better results than context vectors, but that order vectors produce consistently lower results. Combining vectors (context and direction) does not improve the results. By comparison, Jones and Mewhort (2007) report a best result of 57.81% (when combining context and order vectors).

<sup>1</sup>[ftp://ftp.cs.cornell.edu/pub/smart/english.stop](http://ftp.cs.cornell.edu/pub/smart/english.stop)

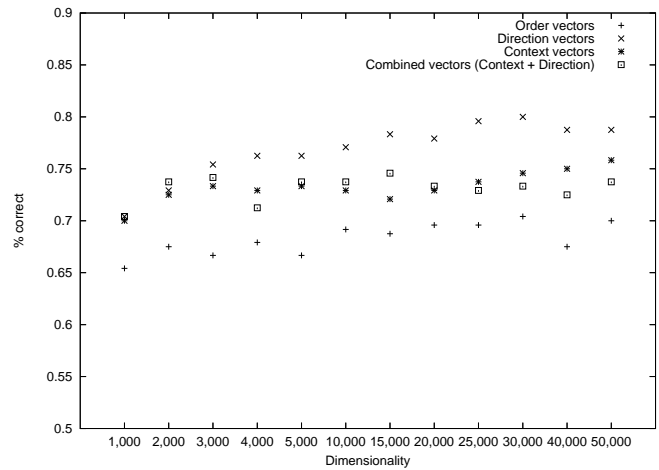


Figure 2: Percent correct answers for different dimensionalities.

### The Effect of Window Size

Figure 3 shows the effect of using different sizes of the context window. In these experiments, we use 4,000-dimensional index vectors with two 1s and two -1s and a frequency cut-off at 15,000. It is obvious that all representations benefit from a narrow context window; 2+2 is the optimal size for all representations except the order vectors (for which a 1+1-sized window is optimal based on TOEFL scores). Notice, however, that with wide windows the order vectors perform better than the others. This might partially explain why Jones and Mewhort see an improvement of the BEAGLE model's performance on the synonym test for the order vectors compared to the context vectors when entire sentences are used as context windows. Unlike in the BEAGLE model, we allow context windows to cross sentence boundaries and have observed decrease in performance when they don't (mean decrease over a large number of tests with different parameters for the direction vectors is 13% and for the order vectors 9%).

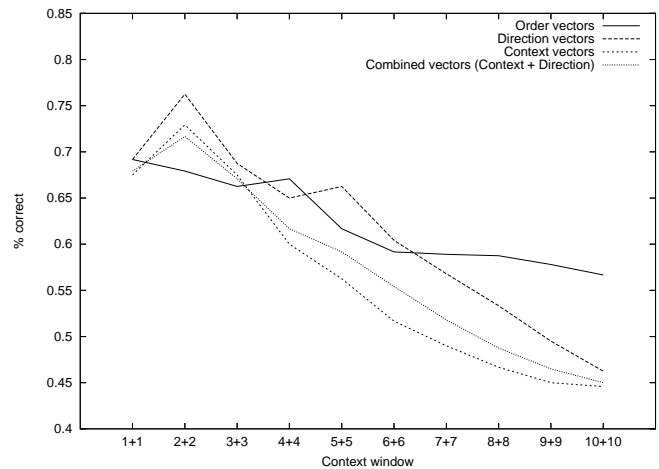


Figure 3: Percent correct answers for different window sizes.

## Discussion

In these experiments, we have used permutations for encoding word-order information into high-dimensional semantic vectors and have achieved gains in a language task that are similar to those achieved by Jones and Mewhort's BEAGLE model. Permutations have the advantage of being very simple to compute. Furthermore, they can be used with any kinds of high-dimensional random vectors, including the ones used by Jones and Mewhort.

Our method of encoding  $n$ -grams with vector *sums* is logically very different from theirs with convolution *products*. When an  $n$ -gram is encoded with a sum of  $n$  permuted vectors, frequent function words in any position overwhelm the sum vector in the same way as in computing context-only memory vectors. We have therefore excluded function words using a cut-off frequency of 15,000 and so our order-based retrieval excludes that information—its import of grammar. The encoding can be modified to include the function words, for example by reducing the weights of very frequent words.

A more significant difference has to do with the specificity of encoding. The convolution product is very specific. For example, the product that encodes “dog bit a man” provides no information on “dog bit the man.” When encoded with a sum (that includes the function words), the two reinforce each other. Furthermore, when using direction vectors, only the order but not the exact position of words matter, giving even more reinforcement between similar but slightly varied sentences. We believe that such similarity in the representation of slightly different ways of saying the same thing gives an increased generalization capability, and explains the good results for the direction vectors compared to order vectors. However, the full relative merits of products and sums, and the best ways of combining them, are yet to be established.

In this and previous studies (Sahlgren, 2006) we have found the optimal context window to be 2+2 when judged by TOEFL scores. It is possible that Jones and Mewhort's context-only memory vectors would be improved by reducing the context window to less than the entire sentence. The encoding of bigrams and trigrams with convolution products may in fact bring about some of the advantages of smaller context windows.

We conclude from these experiments that the permutation of vector coordinates is a viable method for encoding order information in word space, and that certain kinds of order information (i.e. direction) can be used to improve paradigmatic word-space representations. However, our experiments were unable to establish an improvement when using full order information. We believe that further study is necessary in order to fully flesh out the relationships between context, direction and order representations.

*Acknowledgement.* We wish to thank four anonymous reviewers for their constructive critique of our submission.

## References

- Gayler, R.W. (1998). Multiplicative binding, representation operators, and analogy. Poster abstract. In K. Holyoak, D. Gentner, and B. Kokinov (Eds.), *Advances in analogy research* (p. 405). Sofia: New Bulgarian University.  
Full poster at <http://cogprints.org/502/>
- Jones, M. N., and Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114 (1), 1–37.
- Kanerva, P., Kristoferson, J., and Holst, A. (2000). Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd annual conference of the Cognitive Science Society* (p. 1036). Hillsdale, NJ: Erlbaum.
- Karlgren, J., and Sahlgren, M. (2001). From words to understanding. In Y. Uesaka, P. Kanerva, and H. Asoh (Eds.), *Foundations of real-world intelligence* (pp. 294–308). Stanford, CA: CSLI Publications.
- Kaski, S. (1999). Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN'98* (pp. 413–418). IEEE Service Center.
- 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., and Atchley, R. (1995). Semantic and associative priming in high-dimensional semantic space. *Proceedings of the 17th annual conference of the Cognitive Science Society* (pp. 660–665). Hillsdale, NJ: Erlbaum.
- Padó, S., and Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, 33 (2), 161–199.
- Papadimitriou, C., Raghavan, P., Tamaki, H., and Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. *Proceedings of the 17th ACM symposium on the principles of database systems* (pp. 159–168). ACM Press.
- Plate, T. A. (2003). *Holographic reduced representations* (CSLI Lecture Notes No. 150). Stanford, CA: CSLI Publications.
- Redington, M., Chater, N., and Finch, S. (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22 (pp. 425–469).
- Sahlgren, M. (2006). *The Word-Space Model*. Doctoral dissertation, Department of Linguistics, Stockholm University.  
<http://www.sics.se/~mange/TheWordSpaceModel.pdf>
- Widdows, D. (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. *Proceedings of HLT-NAACL 2003* (pp.197–204). Association for Computational Linguistics.