Real, Complex, and Binary Semantic Vectors

Dominic Widdows[1] and Trevor Cohen[2]

[1] Microsoft Bing
[2] University of Texas School of Biomedical Informatics at Houston

Abstract. This paper presents a combined structure for using real, complex, and binary valued vectors for semantic representation. The theory, implementation, and application of this structure are all significant.

For the theory underlying quantum interaction, it is important to develop a core set of mathematical operators that describe systems of information, just as core mathematical operators in quantum mechanics are used to describe the behavior of physical systems. The system described in this paper enables us to compare more traditional quantum mechanical models (which use complex state vectors), alongside more generalized quantum models that use real and binary vectors.

The implementation of such a system presents fundamental computational challenges. For large and sometimes sparse datasets, the demands on time and space are different for real, complex, and binary vectors. To accommodate these demands, the Semantic Vectors package has been carefully adapted and can now switch between different number types comparatively seamlessly.

This paper describes the key abstract operations in our semantic vector models, and describes the implementations for real, complex, and binary vectors. We also discuss some of the key questions that arise in the field of quantum interaction and informatics, explaining how the wide availability of modelling options for different number fields will help to investigate some of these questions.

1 Introduction

The contribution described in this paper is a learning and representation system that enables vector models to be built easily using real, complex, or binary numbers as coordinates for semantic vectors.

Quantum mechanics, statistical machine learning, and hyperdimensional computing have used some notion of state-vector or feature-vector for decades. While these and many other fields use common mathematical vector-space theories, in practice they often differ in their choice of a ground-field, or basic number type. That is, if a vector is a list of coordinates, what sort of numbers should the coordinates be?

In quantum mechanics, and other areas of physics including electromagnetism, complex numbers are indispensable. The Schrödinger equations and Pauli matrices involve complex numbers explicitly, complex numbers are part of the relationship between positions and momenta, and complex Hilbert spaces are

J.R. Busemeyer et al. (Eds.): QI 2012, LNCS 7620, pp. 24–35, 2012.

so normal that the logic of projections in Hilbert space is sometimes called a Standard Logic [1, Ch 1].

Logical semantics and computer science, on the other hand, use mainly binary and set theoretic representations, stemming from George Boole's innovation of describing an Aristotelian predicate as a mapping from a set of objects to the binary numbers [2]. Representations in information theory and modern computing assume an underlying quantized bit-vector, and in software engineering, a 'real number' is really a string of bits mediated by IEEE standards. Standard logic in these fields is Boolean logic.

The growing community of practice in statistical machine learning uses real vectors for most representations. Feature vectors are used to describe items to be classified or ranked, and the features are most often measurable quantities (such as the redness of a pixel in an image, or the weight of a particular term in a given document). This makes idea of using real numbers as features intuitively practical, and standard statistical distributions and techniques are so readily available that the use of 'real' mathematics leaves little to be desired in many successful applications to date [3].

Today, the sciences of intelligence are part of this arena as well. Artificial intelligence and computational linguistics have grown, and partly shifted from an emphasis on binary and logical representations to real and statistical ones. Psychological and cognitive applications of real vectors and their similarities include Prototype Theory [4], Pathfinder Networks [5], and the Conceptual Spaces of Gardenförs [6]. Kanerva's hyperdimensional computing [7] (and of course, Boole's Laws of Thought [8]) use binary representations to model cognitive processes of learning, remembering, and reasoning. The use of complex numbers to model cognitive processes is still apparently in its infancy (see for example [9,10]), but we might well expect this area to grow as well.

Another rapidly growing area is the application of more sophisticated product operations in semantic vector space models. For many decades, the main operations used in vector space models were just vector addition for composition and the cosine or related measures for judging similarity. Though many other operations are well-known in the theoretical literature (for summaries, see e.g., [11,12]), practical implementations have lagged behind, partly for computational reasons. This has changed dramatically over the past few years: several operators have been used successfully in practice to model word-order dependencies [13,14] and operations such as verb-argument binding [15,16], adjective-noun modification [17], and formal ontological relationships [18]. The old notion that distributional semantics is a 'bag-of-words' methodology has been comprehensively superseded, at least in the research literature.

The accelerated development of so many sciences and technologies has naturally left many possible combinations of empirical challenge and mathematical representation unexplored. That is (for example), there are many cognitive models or machine learning tasks to which complex or binary vector representations have not been applied. Of course, if complex numbers, including their so-called 'imaginary' parts, turned out to be a key to modelling mental processes, we

might be surprised and skeptical: but perhaps no more surprised than we should already be at the usefulness of imaginary numbers in electrical engineering.

Some strides have already been made in statistical learning using complex numbers [10] (with theory based on [9]), and with binary numbers [19] (with theory based on [7]). The project undertaken by the authors, and reported in this paper, is to unify these mathematical options in a system that makes it possible to easily experiment with all three (and potentially more) standard ground fields for vector representations. The system is implemented and released in the Semantic Vectors package [20], an open-source package that can be freely downloaded from `semanticvectors.googlecode.com`. For real, binary, and complex vectors, the package now supports training of term and document based semantic models, and makes available a range of product operators for learning similarity, directional, and logical relationships.

This paper is organized as follows. Section 2 describes the mathematical abstractions supported in all representations. Sections 3, 4, and 5 describe the specific operators and implementation decisions for real, complex, and binary vectors respectively. Section 6 discusses the relationship with quantum interaction in more detail. The models used in practice have some significant similarities and differences with the 'classical' quantum mechanical model of complex Hilbert space, and this can be used to shed fresh insight on the important question of what generalizations are appropriate in generalized quantum structures. While this section draws some points in conclusion, readers whose primary interest is in quantum interaction and generalized quantum structures may consider reading Section 6 first. Section 7 briefly refers to experiments conducted using the system. These experiments are described in full in a separate paper.

2 Common Mathematical Operators

This section owes much to the theoretical framework of Kanerva's hyperdimensional computing [7], and the experimental implementation and notation used in [19]. Some of the core concepts are from the literature on Vector Symbolic Architectures (see [9,12,11] and others). Please refer to these papers for detailed motivation: due to space constraints, many of the more subtle points are not discussed in this section.

The goal (as with much of abstract mathematics) is to define a core set of operations that all semantic vector models should support, and rules surrounding these operations. The most basic rules are listed in Table 2. To date, it is better to think of these as rules of thumb, rather than formal axioms that described an algebraic structure such as a group or lattice: such a hardened theory may arise from this work in the future, but it is not yet here.

There are many key discussion points that make these vector systems functionally appealing. In high dimensions, they are easy to build. Randomly allocated elemental vectors are overwhelmingly likely to be unrelated (e.g., pseudo-orthogonal), and large numbers of these elemental vectors can be created before there is any appreciable danger of confusing two vectors. It follows from this that

Table 1. The core mathematical operations performed on representation vectors

- **Generate Random Vector**. Creates a random vector that can be used to represent an elemental concept.
- **Measure Overlap**. Measures the similarity between two vectors A and B: a real number, $A \cdot B$, typically between 0 (no similarity) and 1 (exact match). Negative values are possible. The overlap between two randomly generated elemental vectors should be near to zero (or some other value that means 'no significant overlap').
- **Superpose**. Takes two vectors A and B and generates a third vector $A + B$, such that $A \cdot (A + B)$ and $B \cdot (A + B)$ are relatively large. Superposition is sometimes called *bundling* in the literature.
  - Superpositions can be weighted by any real (in practice, double-precision floating point) number.
  - This, and the presence of a zero vector, gives us the practical ability to perform regular 'scalar multiplication', at least with real number scaling factors.
- **Normalize** Takes a vector A and rescales it to a vector $\hat{A}$ such that $\hat{A} \cdot \hat{A} = 1$.
- **Bind**. Takes two vectors A and B and generates a third vector $A \otimes B$, such that $A \cdot (A \otimes B)$ and $B \cdot (A \otimes B)$ are usually near to zero.
- **Release**. Inverse of bind, written as $A \oslash B$. Should behave in such a way that $(A \oslash (A \otimes B)) \cdot B \approx 1$.

Operator precedence when written is as expected: $+$ comes before $\otimes$ which comes before $\oslash$ which comes before $\cdot$.

superposition is normally quite easy and natural to define (natural in the mathematical sense, that a choice of overlap measure makes some particular choice of superposition operator appealing).

Binding is different: given a choice of overlap measure, there are usually many options for defining an appropriate binding operation. This leaves much freedom for choosing options that are computationally appealing: as we will see in the implementation sections, this is important for building tractable systems. Since many binding operations are available, hybrid systems that use more than one binding operation to represent different semantic combination operations are quite likely to emerge.

Training a model — that is, the process of deriving semantically significant representation vectors from elemental vectors and a training corpus — can then be performed in linear time by taking linear combinations of elemental vectors. Many details of available training processes are available in our earlier works, e.g., [20]. There are several more algorithmically sophisticated training techniques available, including singular value decomposition (see [21] and related literature).

In practice, these core vector operations are declared by a `Vector` interface and implemented by all vector types. Note that the use of an interface (as opposed an abstract base class containing some shared implementation) means that we are making no presuppositions about the physical representation of vectors: in particular, we do not explicitly assume that vectors are lists of coordinates. The implementations so far released in Semantic Vectors are indeed coordinate-based, but coordinate-free representations are not unthinkable.

Other mathematical operations including orthogonalization are supported as utility functions derived from the primitive operations. This allows representations to make use of quantum-logical connectives for search (see [22, Ch 8]). Optimized linear-time search and k-nearest-neighbour ranking are implemented. Each vector implementation is also required to implement common serialization operations, for e.g., writing to long-term storage hardware. In practice, each vector implementation comes with lightweight (often sparse) representations to support large numbers of elemental vectors, and more costly representations for dense semantic vectors.

This concludes our summary of the operations common to all vectors. We will now proceed to describe the three implementations available so far, for vectors using real, complex, and binary numbers as coordinates.

3 Real Vectors

The use of real vectors for representation in empirical learning is by far the most common choice of ground-field to date. In the Semantic Vectors package, real vectors are implemented using single-precision, 4-byte floating point numbers. Randomly-generated elemental vectors are sparse ternary vectors: ternary, meaning that they use only values from the set $\{-1, 0, 1\}$, and sparse, meaning that most values are left as zero. Superposition is implemented using standard component-wise vector addition, and overlap is measured using cosine similarity [22, Ch 5].

For binding, options in the literature include:

- Superposition after permutation of coordinates (introduced by [14]). That is, $A \otimes B$ is implemented by permuting the coordinates of B and then superposing with A. Since there are $n!$ possible permutations, there are $n!$ possible binding operations. The availability of so many options has been used to give different permutations based on the number of words between two terms [14], and to represent different semantic relationships from a knowledge base [19].
- Convolution of vectors, as described in [9]. This was used to model word-order relationships by [13].

Due to computational considerations, the operation used for binding real vectors in the Semantic Vectors package is permutation of coordinates, though an implementation of circular convolution using fast Fourier transforms is available in codebase.

We note in passing that traditional LSA (that is, the creation of a reduced term-document matrix using singular value decomposition) is only available for real vectors.

4 Complex Vectors

The use of complex numbers for semantic representation is discussed in [9] and was first introduced to the Semantic Vectors package in [10]. The extra richness

over real representations comes largely from complex multiplication, which has a an angular or 'turning' effect. This has powerful consequences. For example, since complex multiplication can effectively turn a vector through a right angle, multiplication can turn a cosine similarity of 1 to a cosine similarity of 0. This makes multiplication an effective candidate for the bind operation. The variety of effective options available has encouraged us to implement two different modes for complex vectors: a Cartesian mode where rectilinear options are used by default, and a polar mode where circular operations are used. In more detail, the operations for complex numbers implemented in Semantic Vectors are as follows.

Random elemental vectors have coordinates that are either zero, or elements of the unit circle group $U(1)$ of complex numbers whose modulus is 1. This is an apt generalization of elemental real ternary vectors, since the set $\{-1, 1\}$ is the intersection of the circle group $U(1)$ and the real line. Both sparse vectors (mainly zeros) and dense vectors (all coordinates members of $U(1)$) have been used in practice, and changing this is an easy command-line configuration. As an optimization, a lookup table for sines and cosines of angles is created, and many of the procedures involving complex number multiplication are implemented using addition of keys in this table. Such a key is often called a *phase angle*.

In polar mode, entries remain confined to the unit circle, and normalization is implemented by projecting each complex coordinate (that is, each pair of real coordinates) to the corresponding angle on the unit circle. Of course, this projection is undefined for zero entries. For this reason, we have introduced a zero element in the angle lookup table, with expected the rule that the zero element maps any other value to zero under multiplication.

The main operations in each mode are as follows:

- **Measure Overlap**
  - In polar mode, normalized sum of differences between each pair of corresponding phase angles.
  - In Cartesian mode, the cosine similarity of the corresponding real vectors: in other words, the real part of the standard Hermitian scalar product.
- **Superposition**
  - In polar mode, the weighted average of the corresponding phase angles. This operation is not associative: angles added later in the process have more significance.
  - In Cartesian mode, standard complex vector addition.
- **Normalization**
  - In polar mode, mapping each complex number to the corresponding phase angle.
  - In Cartesian mode, scaling each coordinate so that the sum of the squares is equal to 1.
- **Binding**
  - In polar mode, circular convolution. The key observation here is that, because the representation is already in a phase angle form, it is in the

'frequency domain' and no Fourier transform is necessary for optimization. Thus, circular convolution is simply the addition of phase angles [9,10].
  - In Cartesian form, no such optimization is so naturally available, and permutation of coordinates is used instead.
- **Release** naturally used the inverse of the corresponding bind operations.

Thus the system for complex semantic vectors has the option of treating a complex number as essentially a rectilinear construct, or as essentially a circular construct. These could be combined further by introducing a modulus representation as well as the phase angle representation. We have not done this yet, partly for computational performance reasons, and partly because the virtues of the two representations are still actively under investigation, and we feel that conflating them may be premature.

5 Binary Vectors

The binary vector representation utilized in Semantic Vectors follows the approach originated by Pentti Kanerva, known as the Binary Spatter Code (BSC) [23]. The BSC depends upon hyperdimensional (d on the order of 10,000) binary vectors. As in our other representations, these can be categorized as elemental vectors and semantic vectors, where elemental vectors are randomly constructed so as to be approximately orthogonal to one another, and semantic vectors are generated by superposition of elemental vectors during the training process. However, there are a number of important differences between this and the other representations we have discussed up to this point.

Firstly, distance in the binary space is measured using the Hamming Distance (HD), a count of the number of bits that differ between two vectors (for example, $HD(1001, 0111) = 3$). Orthogonality is defined as a HD of half the dimensionality of the space [24] — a normalized HD of 0.5. This is in keeping with the construction of elemental vectors, which are constructed by distributing an equal number of 1's and 0's at random across the dimensionality of the space. While these vectors are therefore not sparse in the sense of having mostly zero values, the *space* is sparsely occupied in the sense that elemental vectors tend to be far apart from one another. A set of elemental vectors constructed in this manner will have a mean pairwise HD of $\frac{d}{2}$, with a standard deviation of $\frac{\sqrt{d}}{2}$ As the pairwise distances are normally distributed, this implies that in a 10,000 dimensional space, we'd anticipate approximately 99.7 percent of elemental vectors having a HD from one another of between 4700 and 5300. This sparseness of the space confers a level of robustness to the model, as an elemental vector can be distorted considerably while remaining closer to its original self than to any other elemental vector in the space.

Superposition of binary vectors occurs by summing up the number of 1's and 0's in each dimension across all of the binary vectors added. If there are more 1's than 0's in a dimension, the superposition is assigned the value 1. If there

are more 0's, a zero is assigned, and ties are broken at random (this can only occur when an even number of vectors are superposed). So the superposition of 01 and 00 could be either 00 or 01, each with a probability of 0.5. The need to keep track of the votes in each dimension raises an interesting implementation issue, as the memory requirements of retaining an exhaustive voting record for a set of 10,000 dimensional vectors prohibit assigning a floating point number to each dimension, and Semantic Vectors often retains a store of term or document vectors in memory during the superposition process.

Consequently, we have opted for a space-optimized implementation of the voting record, comprising of an ordered array of binary vectors, implemented using Lucene's OpenBitSet class. This implementation also allows for superposition to occur using efficient bitwise operators, without the need for iteration across the O(10,000) dimensions individually, as illustrated in Table 2. This is accomplished by maintaining a temporary 'cursor' vector, of the same dimensionality as the rows, and performing a series of sequential bitwise XOR and NOT operations. The superposition can be weighted by initiating the process at an appropriate level of the voting record, and the size of the voting record can be constrained by maintaining a global minimum value and ensuring that only values beyond this are stored in the record.

Table 2. Space Optimized Superposition. VR = Voting Record. CV = Cursor Vector (initially, the vector to be superposed). $\hat{\text{VR}}$ = altered Voting Record (VR XOR CV). $\hat{\text{CV}}$ = altered Cursor Vector (CV NOT $\hat{\text{VR}}$).

Row	**VR**	**CV**	$\hat{\textbf{VR}}$	$\hat{\textbf{CV}}$
1	1 0 1	1 0 1	0 0 0	1 0 1
2	0 1 1	1 0 1	1 1 0	0 0 1
3	0 0 0	0 0 1	0 0 1	0 0 0
Value	1 2 3		2 2 4	

Once the voting is complete, the value in each dimension is calculated. If this is more than a half of the number of votes in total, a one is assigned to the superposition product in this dimension. If it is less than half, a zero is assigned, and ties are broken at random. The binary vector implementation also facilitates binding, which is an invertible multiplication-like operator that is used to combine vectors with one another. In the BSC [23], elementwise exclusive OR (XOR) is used to accomplish binding. As this operator is its own inverse, is also used to reverse the binding process. It is also possible to accomplish reversible transformation using a permutation, by shifting or swapping the bits of a vector. As elementwise operations on hyperdimensional binary vectors are computationally inconvenient, Semantic Vectors implements permutation 64 bits at a time, by shifting or swapping the elements of the array of *long* integers that underlie Lucene's binary vector (OpenBitSet) implementation.

In addition to the fundamental Vector Symbolic operations of superposition (or bundling) and binding [12], we have implemented binary approximations of orthogonalization, and quantum disjunction [25]. Orthogonality in binary vector space is defined by a HD of a half of the dimensionality of the space. Give two similar vectors, A and B, A can be rendered almost-orthogonal to B by introducing random noise in dimensions that these vectors have in common. Conversely, two vectors with a HD of more than half the dimensionality can be rendered orthogonal by randomly selecting dimensions in which these vectors differ, and changing them accordingly. The binary approximation of orthogonalization facilitates a binary approximation of quantum disjunction. This operator involves transforming the component vectors for disjunction into a subspace made up of mutually orthogonal vectors using the Gram-Schmidt procedure, so that no information is redundantly represented. Subsequently, a vector C can be compared to this subspace by measuring the length of C's projection in the subspace, $\hat{C}$, and comparing this to the length of C ($\frac{\|\hat{C}\|}{\|C\|}$ is the cosine of the angle between C and the subspace). In binary space, we approximate this projection by adding together the normalized HD - 0.5 between the vector C and each of the components of the subspace, to provide a measure of the extent to which the cumulative similarity between the vector C and all of the components of the subspace is greater than what one would anticipate by chance.

6 Typed Vectors and Quantum Interaction

This section examines the mathematical structures we have developed from the point of view of quantum interaction and informatics. Our hope here is that comparing the behaviour of real, binary, and complex vectors will help to answer one of the most pertinent questions in Quantum Interaction: what makes a system "quantum" at all?

Some strong candidate answers to this question have been proposed. Some have suggested that Born's rule for probabilities is key, Aerts *et al.* have concentrated on the Bell inequalities [26], Bruza and Kitto on entanglement [27]. In information retrieval, quantum formalisms are central to the geometric models described by [28] and [22], the former focussing particularly on representing conditionals, and the latter on representing logical connectives.

There is an accompanying debate on how central the quantum properties should be: for example, is it proper to talk more about quantum-like or Generalized Quantum Systems following Khrennikov [29]. The situation is complicated by history: several properties of vector spaces (especially the logic of projection onto subspaces) were explored in the service of quantum mechanics, and even with hindsight it is not always easy to say which parts of the theory are necessarily quantum, and which would be better described as properties of all vector space models.

One property common to all our high-dimensional models is the sparseness of point distribution. Semantic vectors are not all sparse in the sense of having most coordinates equal to zero, but they are sparse in the sense of being very

spread out in their spaces: there are very few actual elemental or semantic vectors compared with the number of points available. Thus, even those spaces that are continuous in theory (such as those based on real and complex numbers) are sparse and thus highly "quantized" in practice.

Thus, many of the quantized properties of semantic vector models arise generally, and not because of any special parallel with the quantum *mechanical* model of complex Hilbert space. For example, there is no preference yet discovered for self-adjoint operators in semantic vector models.

Here, our work has a particular contribution to make. We can now do many experiments in textual informatics while comparing complex, real, and binary vector space models. If a strong correspondence with quantum mechanics itself actually exists, we would expect to see complex Hilbert space representations to be distinctly superior for at least some modelling operations.

If, on the other hand, real or binary representations work best, we may come to the more guarded conclusion that our semantic models are (obviously) vector models, as are quantum Hilbert space models, but the similarity between semantic models and quantum mechanics goes no further. Such findings would even further motivate the question "What characterizes quantum systems?", rather than especially quantum *mechanical* systems. Consider the way the voting record for binary vectors is maintained during learning and quantized to binary values during normalization: the practical benefits of quantization itself are clearer in the binary representation than in either the real or complex representation. Many 'quantum' properties (such as entanglement and non-commutativity of observables) may be formulated with vectors and matrices over any ground field.

We note also that there our particular use of Vector Symbolic Architectures, which owes much to [9] and [7], is a specific class of vector models in which product operations yield other vectors. The mathematical options are of course much richer and sometimes demand more computational resources: for recent and empirically successful examples, see the use of tensor products [16] and matrices by [17].

7 Experiments and Evaluation

Using the system described in this paper, detailed experiments (using especially complex and binary vectors) have already been conducted on using multiple cues for analogical reasoning. These experiments use quantum disjunction to model the many possible relationships that could be used as premises to deduce new semantic relationships. Due to space constraints, these experiments are not described here but in a separate paper [30]. These experiments also introduce an innovative notion of 'binary quantum disjuction', which is altered from the standard real or complex version, partly because the average similarity between unrelated binary vectors is the normalized Hamming distance of 0.5. Of course, these are (we hope) the first of many experiments that will compare and adapt real, complex and binary techniques.

What we do present in the current paper is an appropriate computational framework (if you will, a laboratory) in which such experiments will be conducted.

We sincerely hope that comparisons between different ground fields for semantic representation will become the norm rather than the exception. If so, this will catalyze technological progress and lead to core scientific insight.

Acknowledgments. This research was supported in part by the US National Library of Medicine grant R21LM010826.

References

1. Varadarajan, V.S.: Geometry of Quantum Theory. Springer (1985)
2. Boole, G.: The Mathematical Analysis of Logic. Macmillan (1847) (republished by St Augustine's Press, 1998, introduction by John Slater)
3. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Springer Series in Statistics (2001)
4. Rosch, E.: Principles of categorization. In: Collins, A., Smith, E.E. (eds.) Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence, Kaufmann, San Mateo, CA, pp. 312–322 (1988)
5. Schvaneveldt, R.W.: Pathfinder Associative Networks: Studies in Knowledge Organization. Intellect Books (1990)
6. Gärdenfors, P.: Conceptual Spaces: The Geometry of Thought. Bradford Books MIT Press (2000)
7. Kanerva, P.: Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. Cognitive Computation 1(2), 139–159 (2009)
8. Boole, G.: An Investigation of the Laws of Thought. Macmillan (1854) Dover edition (1958)
9. Plate, T.: Holographic Reduced Representations: Distributed Representation for Cognitive Structures. CSLI Publications (2003)
10. de Vine, L., Bruza, P.: Semantic oscillations: Encoding context and structure in complex valued holographic vectors. In: Proceedings of the AAAI Fall Symposium on Quantum Informatics for Cognitive, Social, and Semantic Processes, QI 2010 (2010)
11. Aerts, D., Czachor, M., De Moor, B.: Geometric analogue of holographic reduced representation. Journal of Mathematical Psychology 53(5), 389–398 (2007)
12. Gayler, R.W.: Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. In: Slezak, P. (ed.) ICCS/ASCS International Conference on Cognitive Science, pp. 133–138. University of New South Wales, Sydney (2004)
13. Jones, M.N., Mewhort, D.J.K.: Representing word meaning and order information in a composite holographic lexicon, vol. 114, pp. 1–37 (2007)
14. Sahlgren, M., Holst, A., Kanerva, P.: Permutations as a means to encode order in word space. In: Proceedings of the 30th Annual Meeting of the Cognitive Science Society (CogSci 2008), Washington D.C., USA, July 23-26 (2008)
15. Widdows, D.: Semantic vector products: Some initial investigations. In: Proceedings of the Second International Symposium on Quantum Interaction (2008)
16. Grefenstette, E., Sadrzadeh, M.: Experimental support for a categorical compositional distributional model of meaning. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP (2011)

17. Baroni, M., Zamparelli, R.: Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP (2010)
18. Cohen, T., Widdows, D., Schvaneveldt, R., Rindflesch, T.: Logical leaps and quantum connectives: Forging paths through predication space. In: Proceedings of the AAAI Fall Symposium on Quantum Informatics for Cognitive, Social, and Semantic Processes, QI 2010 (2010)
19. Cohen, T., Widdows, D., Schvaneveldt, R., Rindflesch, T.: Finding Schizophrenia's Prozac Emergent Relational Similarity in Predication Space. In: Song, D., Melucci, M., Frommholz, I., Zhang, P., Wang, L., Arafat, S. (eds.) QI 2011. LNCS, vol. 7052, pp. 48–59. Springer, Heidelberg (2011)
20. Widdows, D., Cohen, T.: The semantic vectors package: New algorithms and public tools for distributional semantics. In: Fourth IEEE International Conference on Semantic Computing, ICSC (2010)
21. Landauer, T., Dumais, S.: A solution to plato's problem: The latent semantic analysis theory of acquisition. Psychological Review 104(2), 211–240 (1997)
22. Widdows, D.: Geometry and Meaning. CSLI Publications (2004)
23. Kanerva, P.: Binary spatter-coding of ordered k-tuples. In: Artificial Neural Networks, ICANN 1996, pp. 869–873 (1996)
24. Kanerva, P.: Sparse distributed memory. The MIT Press, Cambridge (1988)
25. Widdows, D., Peters, S.: Word vectors and quantum logic. In: Proceedings of the Eighth Mathematics of Language Conference, Bloomington, Indiana (2003)
26. Aerts, D., Aerts, S., Broekaert, J., Gabora, L.: The violation of bell inequalities in the macroworld. Foundations of Physics 30, 1387–1414 (2000)
27. Galea, D., Bruza, P., Kitto, K., Nelson, D., McEvoy, C.: Modelling the Acitivation of Words in Human Memory: The Spreading Activation, Spooky-activation-at-a-distance and the Entanglement Models Compared. In: Song, D., Melucci, M., Frommholz, I., Zhang, P., Wang, L., Arafat, S. (eds.) QI 2011. LNCS, vol. 7052, pp. 149–160. Springer, Heidelberg (2011)
28. van Rijsbergen, K.: The Geometry of Information Retrieval. Cambridge University Press (2004)
29. Khrennikov, A.: Ubiquitous Quantum Structure: From Psychology to Finance. Springer (2010)
30. Cohen, T., Widdows, D., De Vine, L., Schvaneveldt, R., Rindflesch, T.C.: Many Paths Lead to Discovery: Analogical Retrieval of Cancer Therapies. In: Busemeyer, J.R., Dubois, F., Lambert-Mogiliansky, A. (eds.) QI 2012. LNCS, vol. 7620, pp. 90–101. Springer, Heidelberg (2012)