Semantic Vector Products: Some Initial Investigations

Dominic Widdows

Google, Inc.

widdows@google.com

To appear in *Second AAAI Symposium on Quantum Interaction*, Oxford, 26^{th} – 28^{th} March 2008

Abstract

Semantic vector models have proven their worth in a number of natural language applications whose goals can be accomplished by modelling individual semantic concepts and measuring similarities between them. By comparison, the area of semantic compositionality in these models has so far remained underdeveloped. This will be a crucial hurdle for semantic vector models: in order to play a fuller part in the modelling of human language, these models will need some way of modelling the way in which single concepts are put together to form more complex conceptual structures.

This paper explores some of the opportunities for using vector product operations to model compositional phenomena in natural language. These vector operations are all well-known and used in mathematics and physics, particularly in quantum mechanics. Instead of designing new vector composition operators, this paper gathers a list of existing operators, and a list of typical composition operations in natural language, and describes two small experiments that begin to investigate the use of certain vector operators to model certain language phenomena.

Though preliminary, our results are encouraging. It is our hope that these results, and the gathering of other untested semantic and vector compositional challenges into a single paper, will stimulate further research in this area.

Introduction and Background

Compositionality in Mathematical Semantics

The work of George Boole and Gottlob Frege in the 19^{th} century pioneered a new era of research and development in mathematical linguistics: that is, the application of mathematical and logical methods to the study and representation of human language. Grandees of this field in the 20^{th} century include Tarski, Montague, Fodor: a survey of this huge area is beyond the scope of this paper, though an interested reader should perhaps begin with Partee, ter Meulen, and Wall (1993). Significantly for the purposes of this paper, the point we would like to emphasize is that this tradition has been strongly influenced by a maxim that has become known as "Frege's context principle" or "Frege's principle of compositionality", which may be translated as

> It is enough if the sentence as whole has meaning; thereby also its parts obtain their meanings.
>
> (Frege 1884, §60); see also (Szabó 2007).

Following this tradition, is has become much more normal for mathematical linguists to give account of *compositional* semantics (the meaning of phrases and sentences) while paying little attention to *lexical* semantics (the meaning of individual words). For example, a typical formal representation of the verb "loves" may use the lambda calculus expression

$$\lambda x\ \lambda y\ [\texttt{loves}(x, y)]. \tag{1}$$

This tells us how "loves" behaves as a transitive verb with respect to its arguments, and allows us to fill in the variables x and y to bind the predicate to these arguments. But it tells us nothing about the difference between "loves" and "hates" or even "eats" and "pays": in short, it tells us absolutely nothing about what it means to be loved, except that there must be something being loved and something doing the loving. One noted problem with filling this gap is the acceptance that the lexicon is much more empirical than the grammatical structure of a language, and so much harder to formalize.

Semantic vector models

One of the great virtues of semantic vector models is that they have begun to fill this lexical gap, by appealing to the empirical data itself. Natural language has not grown any easier to model formally, but it has become extremely easy to get hold of and to process large amounts of the raw phenomenon itself, in the form of written documents available electronically. By the early 1990s, this had lead researchers to create large scale models of words and their relationships, initially by factoring large term-by-document matrices as used in information retrieval systems (Deerwester *et al.* 1990; Landauer & Dumais 1997) and later by examining cooccurrence between pairs of words in textual windows (Lund & Burgess 1996; Schütze 1998).

Table 1: Words similar to "loves" in a semantic vector model constructed from the British National Corpus.

Word	Score
loves	1.000000
love	0.702550
loving	0.613473
tells	0.524047
loved	0.514558
god	0.491559
bless	0.483682
spirit	0.482846
thee	0.479075
hate	0.478082
believer	0.474993
intimacy	0.472540
passionate	0.467109
affections	0.458393

Semantic vector models are models in which concepts are represented by vectors in some high dimensional space W (usually $100 < \dim W < 500$). Similarity between concepts may be computed using the analogy of similarity or distance between points in this vector space. Semantic vector models are by now a recognized part of mainstream computational linguistics, and are sometimes described as WORDSPACE models (Widdows 2004; Sahlgren 2006). Their formal theory as part of information retrieval, and the relationship of their logic to that of quantum mechanics, is also well established (van Rijsbergen 2004). Their importance is also becoming established in cognitive science, championed by Gärdenfors (2000) as "Conceptual Spaces". There is considerable cognitive significance in the fact that semantic vector models can be learned and represented using only sparse distributed pieces of memory such as might be available in individual neurons (Kanerva 1988). Thus, there is a potential account for how such models may take shape in the brain, something that is still largely lacking for formal symbolic models. It is also interesting that this method for constructing semantic vectors, known sometimes as "Random Indexing" or "Random Projection" is also the most computationally tractable and thus likely to lead to commercial-scale applications (Sahlgren 2005; Widdows & Ferraro 2008).

Strengths and Weaknesses, and the Goals of this Paper

To get a sense of the complementary strengths of semantic vector and lambda calculus representations, compare the table of similar terms to "loves" in Table 1 with the lambda calculus expression in Equation (1).

The two representations are clearly doing very different things, and it is easy to see that both are important parts of understanding what the word "loves" means and how it may be used in English. In a larger survey, other models could well be included such as a (smoothed) collection of n-grams (Manning & Schütze 1999, Ch. 6), relations in a lexical network such as WordNet (Fellbaum 1998), and a traditional dictionary definition. The two representations we have considered explicitly are stressed here because they motivate the central question of this paper: is it possible to have a semantic calculus that can exhibit both the compositional sophistication of traditional formal models like lambda calculus, and the empirical robustness and basic associational abilities of a semantic vector model?

This paper does not provide a full answer to this question — as we will demonstrate, it is not one question but many. Instead, we attempt to summarize what we believe to be the main challenges and opportunities posed by the question, and describe a small handful of initial experiments. The work builds on that of other researchers, particularly Plate (2003) and Clark and Pulman (2007). Our main intent is to demonstrate that, in spite of obvious setbacks and pitfalls, there are many possible avenues of exploration which between them contain sufficient promise of useful results that much further research should be done in this area.

Semantic Product Operations

There are many many forms of semantic composition in natural language. Some of these have received a great deal of attention in the literature of mathematical linguistics, and others have been less thoroughly studied. In this section, we present a summary of a few of the more typical composition operations, and give some idea of systems in which they are important.

Functional Composition

In many syntactic frameworks, verbs are modelled as functions that take nouns as their arguments. The lambda calculus expression in Equation (1) is typical. We describe this as functional composition in the mathematical or computational sense: the verb is modelled as a "function" that takes two "variables" (nouns) and returns a statement or sentence. In truth-functional accounts, the composed meaning of the statement, once the variables have been put in the right slots, is the set of worlds or situations in which the statement is true. Functional composition is naturally important for natural language processing systems that are trying to map natural language statements onto some fragment of the real world, such as task-oriented dialogue systems. The modelling of functional composition in language is often traced to Frege's formulation of predicate calculus, though the notation we use today owes more to the American philosopher C. S. Peirce.

Morphological Inflection

In English, the sentence "She speaks." is a functional composition of a pronoun with an intransitive verb. However, in many languages this meaning is produced as a single word. For example, in Spanish the verbal

root "habl-" is inflected with the ending "-a" so that the single word "habla" may convey the same meaning. Though not usually thought of as a topic in semantic composition, computational morphology is clearly related and needs to be part of language processing systems that involve any level of language parsing and generation.

Logical Connectives

Half-a-century before the work of Frege, Boole introduced the algebra of logical connectives, and specifically intended them to be used to model statements in natural language (Boole 1854). All computer scientists and logicians are familiar with Boolean logic, which has become the propositional core of the more complex predicate calculus. As such, Boolean logic is available in every modern computer programming language, whereas other logical operations such as quantifiers often have to be coded by hand, partly because the computational complexity and time/space tradeoffs of these operations varies considerably from situation to situation.

The most obvious examples of logical connectives in user-facing natural language systems is in traditional keyword search engines, still available as part of the "Advanced Search" interface in Internet search engines such as Google.

Complex Nominals

Traditional texts on mathematical linguistics often lead the student to believe that a functional approach based on predicate logic is the correct way to model the basics of semantics in natural language, leaving a host of other issues to be treated as more advanced topics (for example, see Jurafsky and Martin (2000, Ch 14)).

Complex nominals are one of the several notable cases where this approach fails. Boolean conjunction being equivalent to set intersection, a "tiger moth" is not a Boolean conjunction of a tiger with a moth, a "tiger economy" is not a Boolean conjunction of a tiger with an economy, and "Romeo and Juliet" is not a Boolean conjunction of Romeo with Juliet. In the latter case, the meaning of "and" is more like the traditional disjunction or union operation, and in the first two cases, the noun "tiger" is being used as a modifier, in the first case to signify "striped" in the second case to signify "fierce".

Adjectival modifiers show similar plasticity (Gärdenfors 2000, p. 120). "Red apples" can be interpreted as a Boolean conjunction, but the meaning of "red wine" (more like the colour purple) and "red skin" (more like the colours pink and brown, perhaps) are both contextually scoped by the space of possible wine and skin colours, of which the listener is presumably aware. "Red politics" and "red herring" go further into the idiomatic realm, though the first involves some compositionality in that the modifier "red" is still selecting a kind of "politics".

Systematic Ambiguity in Compositions

As alluded to above, many modifiers take on different meanings depending on the head-word they are modifying. For example, a "long house" has a large physical length, but a "long flight" takes a long time. (One might argue that this is still a function of length, but then one must account for the fact that a single distance may be covered by a long drive or a short flight.) Still further, a "long book" takes a long time to read, and a "long light" is a traffic light that takes a long time to change. These semantic phenomena are well known, and are described particularly by Pustejovsky (1995).

Summary

We have only listed a few types of compositional operations in natural language, but it should already be clear that the functional and propositional operators studied at length in mathematical linguistics do not account for all of them, and that semantic composition in natural language cannot as a whole be modelled using a single mathematical operation, or even using handful of operations within a single mathematical structure (at least, not one that has yet been discovered).

In practical language processing systems, this has left many gaps to be filled by largely informal or accidental means. For example, we expect a search engine to return the correct documents for the query "tiger moth," not because the Boolean model correctly composes the meanings of the individual terms, but because the Boolean model picks out documents whose authors have correctly composed these meanings (where by "correctly" we mean "according to the convention that the user expects"). It is easy to see situations where this doesn't serve user needs. For example, a user may encounter a tiger moth for the first time, and not knowing its name, may produce the query "moth with black and orange stripes." Again, a search engine may respond with informative documents, but only if one of the authors has written a document that conveniently says words to the effect that "This moth with black and orange stripes is called a tiger moth." We are still relying on the users, not the system, to have correctly designed and interpreted the natural language descriptions, and as we all know, this does not always work very well. This is not to say that natural language processing does not have some tools to offer: for example, collocation extraction tools (Smadja 1993; Manning & Schütze 1999) may be used to help with extracting nounphrases and indexing them accordingly. But such tools are rarely semantic in nature: they will discover that a particular phrase is more-or-less fixed, but will not yet infer any special relation between this lexical fixedness and a conventionally accepted meaning.

Vector Space Product Operations

We would like to be able to use models in which the semantics of individual elements gives rise to seman-

tic interpretations of compound expressions. This section describes some of the operations that are available for composition in vector space models. Again, we are forced to be brief and refer readers to more detailed textbooks in linear and multilinear algebra for more detail (Jänich 1994; Fulton & Harris 1991).

Vector Addition

The simplest operation for composing two vectors v and w in a vector space W is the vector sum. This is a mapping $W \times W \to W$ which gives the vector space its basic structure as a commutative group. Vector addition is used throughout vector space approaches to information retrieval to compute vectors for documents and multiword queries from vectors for individual terms (Salton & McGill 1983; Baeza-Yates & Ribiero-Neto 1999). Often terms in the sum are weighted using some measure of information content such as *tf-idf*. Vector addition does perform adequately enough to get many operations to work in information retrieval systems, in spite of the obvious objection that due to commutativity, such systems do not distinguish a *blind Venetian* from a *Venetian blind*.

Vector addition has been used for operations other than document retrieval, particularly word sense discrimination (Schütze 1998). Word sense discrimination first uses clustering of word vectors to obtain clusters that can be interpreted as word senses, uses vector addition of terms in a given context window to form a context vector for each occurrence, and then assigns each occurrence to a word sense using cosine similarity. For an introduction to vector addition, cosine similarity and clustering, see (Widdows 2004, Ch. 5,6).

Direct Product

The direct product $V \oplus W$ of two vector spaces V and W is the vector space formed of elements $\{v \oplus w \mid v \in V, w \in W\}$, with addition defined by the identity

$$v_1 \oplus w_1 + v_2 \oplus w_2 = (v_1 + v_2) \oplus (w_1 + w_2).$$

Direct products are additive in dimension, that is, $\dim(V \oplus W) = \dim V + \dim W$. Direct products may be seen as a way of combining vectors in a way that keeps the identities of the constituents separate.

Logical Connectives and Quantum Logic

The first attempts to incorporate something akin to Boolean operators into vector models for information retrieval was in the work of Salton, Fox, & Wu (1983). This work uses p-norms to normalize a sum of query terms, defined by

$$L_p(x) := \left(\sum_{i=1}^{n} x_i^p \right)^{1/p}.$$

The notion is that, by tuning the parameter p, a query score can be made to select more specifically for similarity along particular dimension (like a conjunction), or can be made to be more tolerant of several near misses in several dimensions (like a disjunction).

The development of a full logic for information retrieval, along with the appreciation that it involves some distinctly non-classical operations, is thanks largely to van Rijsbergen (1986 and ongoing), culminating recently in the thorough demonstration that the vector space logic for information retrieval is the same as the quantum logic of Birkhoff and von Neumann (1936). (See (van Rijsbergen 2004) and also (Widdows 2004, Ch 7).)

In any vector space W, the subspaces of W naturally form a lattice in which the meet of two subspaces U and V is given by their pointwise intersection $U \cap V$, and the join operation given by their linear sum $U + V = \{u+v \mid u \in U \text{ and } v \in V\}$. If neither subspace contains the other, this operation introduces many points that are in neither of the original subspaces, which is why the lattice is *non-distributive*, which formally accounts for the possibility of a particle being in a mixture of states in quantum mechanics.

If the vector space W is also a Hilbert space (complete, and equipped with a Hermitian scalar product), then the space has well-defined orthogonal projection and orthogonal complement operations. In practical examples of WORDSPACE, this is always the case, the Hermitian scalar product being the standard scalar product from which cosine similarity is defined. The operations of projection onto an intersection, a linear sum, and an orthogonal complement give respectively the conjunction, disjunction and negation operations of a logic. In a confusion of terminology, many computer scientists and mathematical linguists refer to this non-distributive structure as a "non-standard logic", whereas mathematical physicists explicitly refer to logics derived from lattices in Hilbert spaces as the "standard logics" (Varadarajan 1985).

Simple consequences arise from these mathematical underpinnings in the ability of the respective logics to accurately represent linguistic meaning. Consider, for example, the sentence

> Catch bus 41 or 52, the journey should take 20 or 25 minutes.

Boolean logic correctly models the discrete statement "Catch bus 41 or 52", which does not imply that any other bus with a number between 41 and 52 is appropriate. Quantum logic correctly models the continuous statement "the journey should take 20 or 25 minutes," for which a journey time of 22 minutes is perfectly consistent.

In semantic vector models, quantum negation / orthogonal projection has (to our knowledge) been evaluated on a larger scale empirical task than any vector composition operation apart from the vector sum. The empirical task in question was the removal of unwanted keywords and their synonyms in a document retrieval system. Quantum negation significantly out-

performed Boolean negation at removing search results that contained synonyms and neighbours of unwanted query terms (Widdows 2003).

General Projection Operators

Quantum logic is a logic of projection operators. While the lattice of subspaces and orthogonal complements is perfectly well defined, it is the associated lattice of orthogonal projections onto these subspaces that is used to model "quantum collapse" and which exhibits the non-commutativity of operators from which Heisenberg's Uncertainty Principle arises.

As a generalization, we should note that projection operators can be considered quite independently of quantum logical connectives. Some researchers have already considered using projections to model conceptual or language operations. For example, Gärdenfors (2000) uses projection onto a convex region to account for the use of the term "red" in "red wine", "red skin", etc. to refer to colours that are not really "red". Widdows (2004) also suggests that projections may be used to model the way a phrase like "tiger moth" selects the appropriate feature of "tiger" as a modifier to the "moth" concept.

Support Vector Machines and Kernel Methods

Though the notion of representing phenomena such as words using vectors is still considered novel by many linguists, it is taken for granted in standard textbooks on statistical learning (Hastie, Tibshirani, & Friedman 2001). In the basic formulation of the problem, it is presumed that the input data consists of a collection of objects described by characteristic feature-vectors (that is, statements of the extent to which an object exhibits a particular feature). The objects may also be labelled, for example, if the objects are text documents, they may be labelled as "spam" or "not-spam", or as "relevant" or "non-relevant" to a particular query. Inferring the correct labels or classification for hitherto unseen objects can then be formulated as a statistical machine learning problem, for which many solutions have been developed.

Support Vector Machines (Cristianini 2000) constitute one family of techniques that is of particular interest to the current discussion. Support Vector Machines work by finding a hyperspace (that is, a subspace of $dim(n-1)$ of the original vector space) that linearly separates the training data into positive and negative examples.

This is an important contribution because it represents concepts (in this case, classes of negative and positive examples) as *regions* in the vector space, not just as individual points. Another way of obtaining regions in the space is to build them using quantum disjunctions. As semantic vector models approach maturity, it will be crucial to solve this problem of *regional representation*, so that (for example) we can model the fact that apples are a kind of fruit, rather than just modelling the fact that apples and fruit have something to do with one another.

Tensor Products

One objection that theoretical linguists may have to all of the above methods is that they give ways of measuring similarity, ranking, or classifying *individual objects*. But language is known to have considerable hierarchical and recursive structure. How could one possibly represent something like a traditional parse-tree as a single point or region in a vector space?

Tensor products and multilinear algebra are one answer to this question. The tensor product may be defined in terms of multilinear maps (Fulton & Harris 1991, Appendix A), though a simpler initial intuition can be obtained by a matrix description: if the vector x has coordinates x_i and the vector y has coordinates y_j then the tensor product $x \otimes y$ is represented by the matrix whose ij^{th} entry is $x_i y_j$ (Plate 2003, §2.4.3). For those more familiar with the Dirac bra-ket notation, this tensor wold be written $|x\rangle\langle y|$, where $\langle y|$ is the adjoint vector of $|y\rangle$.

Just as there are matrices which cannot be formed by taking the outer product of two individual vectors, there are tensors that cannot be obtained as the tensor product of two individual vectors. This gives rise to a formulation of the phenomenon known as quantum entanglement (Rieffel 2007).

Tensor products are comparatively little known in computational linguistics and artificial intelligence, though their use was initially advocated by Smolensky (1990). Recent interest in tensor products includes the work of Clark and Pulman (2007), who propose the use of tensor products to combine the benefits of symbolic and distributional models of meaning, very much the research goal we are following here. As pointed out by Rieffel (2007), tensor products are as much a part of classical probability theory as they are of quantum theory, since they are the correct space for forming the joint probability distribution of two distributions.

Tensor products can be composed recursively, and can therefore be used to model hierarchical structures. This has been demonstrated in a small artificial WORDSPACE by Aerts and Czachor (2004).

Convolution Products and Holographic Reduced Representations

One of the objections to the use of tensor products (and equally for joint probability distributions) is that the number of dimensions increases exponentially with the number of spaces joined. It is reasonable that a semantic representation of a sentence should consume more space in memory than that of a single term, and that a long document should take more space than a single sentence. However, there are good practical and theoretical reasons for believing that this representation should scale no more than linearly in the length of the document.

A variety of product operations have been proposed that are effectively compressed tensor product operations. The most thorough summary is given by Plate (2003) in his description of *Holographic Reduced Representations*, or HRRs. A predecessor of HRRs is the *convolution product*, formed by adding the diagonals of a matrix representation of a tensor product. If v and w are of dimension n, it follows that $\text{conv}(v, w)$ is of dimension $2n + 1$. HRRs are formed by taking this projection a step further: each diagonal is wrapped round, effectively summing the k^{th} and $k + n^{th}$ entries of the convolution product to give a HRR product of dimension n. Plate used HRRs to model semantic composition in a number of simulated and hand-built experiments.

Two Small Experiments

In this section we present two experiments in the use of vector composition to model semantic composition. Though the experiments themselves are small, they were performed using vectors from large empirical semantic vector spaces built using the 100 million word British National Corpus[1] and the Infomap NLP software[2], which uses singular value decomposition to create a reduced WORDSPACE from a large cooccurrence matrix (Widdows 2004, Ch 6). The number of dimensions used in the reduced space was 100. For clarity, we should stress that the word vectors used in these experiments were hand-picked for investigation (though they were not cherry-picked for good results). There is no claim as yet that these results are representative or immediately generalizable. The methods described here still leave much scope for empirical exploration, and we are not yet in a position (nor do we wish) to fasten our colours to a particular statistical evaluation.

Relations between Cities and Countries

The first experiment investigated the use of vector product operations to encode relationships between cities and the countries that contain them. As a test dataset, vectors for 10 capital cities and the corresponding countries were collected, giving 20 vectors altogether. The experiment attempted to produce some kind of query expression and similarity function so that a capital city could be used as a query and retrieve the country it is in. As a baseline case, the other 19 vectors were sorted by their cosine similarity with the test city. As a compositional test, the query was primed with the seed relation *moscow - russia*. That is, instead of just finding vectors similar to a city v, we find vectors w whose relationship with v is similar to the relationship between *moscow* and *russia*. The compositional methods tested were the tensor product and convolution similarities, using the naturally induced similarity functions on these objects.

The results were varied. For many of the capital / country pairs (e.g., *berlin - germany*, *dublin - ireland*, *washington - usa*, the straight (unprimed) vector similarity was the strongest association. However, in some cases the straight vector similarity was not so strong, particularly in the case of *london - britain*. In this case, *britain* was 13^{th} out of 20 in the results. We suspect this is because *london* and *britain* both occur frequently in the British National Corpus, in contexts where they are not strongly associated with each other (i.e., where one term occurs and the other does not). For the task of using the query *london* to find the result *britain*, the tensor product ranking moved *britain* up to the fourth place in the list, and the convolution product ranking moved it straight to the top. This result proved to be quite reliable for the tensor product, in that seeding the search with other relation pairs (e.g., *paris - france*) demonstrated similar improvement. The convolution product appeared less predictable, with some seed relations producing better results and some producing worse ones.

One preliminary conclusion from this experiment (which we may have expected) is that for rarely occurring and strongly correlated terms, simple vector similarity is likely to associate them strongly. However, for relatively common terms with a relationship that is not always expressed, using a seed relation to prime a search that uses a vector composition operation may be more effective. This may be a new lead in the challenge proposed by Bruza et al. ((Bruza & Cole 2005), (Widdows & Bruza 2007)) of predicting from the medical literature that the relationship between *Reynaud's syndrome* and *fish oil* was significant, in spite of their lack of immediate cooccurrence.

It is also important to note that building a tensor product $|v\rangle\langle w|$ using a single relationship between $|v\rangle$ and $|w\rangle$ gives an unentangled representation, so unless we use more than one training example, the results can be expressed as products of individual scalar products (as described by Clark and Pulman (2007), see also the next experiment below).

Similarity Between Verb-Noun Pairs

This experiment was designed to compare the sensitivity of different product operations and similarity measures for modelling functional composition, particularly the composition of verb-noun pairs, where the verb was a transitive verb and the noun was in the object position / patient role of the verb. The intuition behind this experiment is that, for many verb-noun pairs, changing one of the constituents completely changes the meaning or intent of the phrase. Even if one argument remains unchanged, the resulting phrase may be completely different or even nonsensical, even though the two phrases may be deemed to be at least "50% similar" in a superficial sense. As a simple example, consider the sentences:

(i.) I eat an apple.
(ii.) I eat a tomato.
(iii.) I throw an apple.

[1] `http://www.natcorp.ox.ac.uk/`
[2] `http://infomap-nlp.sourceforge.net/`

Whatever the individual similarities between *eat*, *throw*, *apple* and *tomato*, we would like to be able to predict that *eating a tomato* is relatively similar to *eating an apple*, whereas *throwing an apple* is quite different.

Since the BNC comes with part-of-speech tags provided by the CLAWS tagger (Leech, Garside, & Bryant 1994), it is easy to build a WORDSPACE which distinguishes between words with the same orthographic form but different parts of speech (e.g., fire_nn1 and fire_vvi). This was done, and vectors for a handful of verbs and nouns were collected as shown in Table 2. The words were then arranged into verb-noun pairs (with some noun-verb pairs as an extra test).

Composed vector expressions were created for each of these pairs, using the vector sum, the direct product, and the tensor product. The natural similarity operation on the direct product is defined by $v_1 \oplus w_1 \cdot v_2 \oplus w_2 = v_1 \cdot v_2 + w_1 \cdot w_2$ (we divided by 2 to take the average). The natural similarity operation on non-entangled tensor products, $(v_1 \otimes v_2) \cdot (w_1 \otimes w_2)$, can be written as the product of the similarities of the constituents, $(v_1 \cdot w_1) \times (v_2 \cdot w_2)$. As is standard with multiplying similarity scores, this guarantees that weak links in the chain have a great effect on the outcome.

The results in Table 2 show that the tensor product representation / product of similarity measures does the best job at recognizing unlikely combinations. For example, both of the other two measures use the strong similarity between *earn_vvi* and *pay_vvi* to infer that *earning money* and *getting paid in apples* have a lot in common: the tensor product similarity recognizes these as being completely dissimilar. (The raw scores themselves are not necessarily comparable between columns, since we have made no attempt to normalize the measure with respect to one another: the significant data is in the comparison of scores in the same column for different rows.)

Note that this experiment is biased to some extent: we know in advance that the simple vector sum is a commutative operation, and the direct sum and tensor product are not, and so we already know that the vector sum is going to perform poorly on any test where the similarity measure is expected to notice that the order of inputs has been changed. However, we can say with some confidence that, in situations where the words in a sentence can be tagged and perhaps assigned to different semantic roles, a similarity between tensor products is likely to represent linguistic similarity more faithfully than one that performs a single "bag of words" vector sum. Since part of speech tagging is today standard, and semantic role labelling is being intensively studied, these premises are becoming increasingly reasonable.

Conclusions and Future Work

Modelling composition of meaning is a crucial challenge in natural language processing. There are many compositional operations in language, and any successful project should pay close attention to which of these operations is being modelled.

Given the success of semantic vector models at many NLP tasks, it would seem particularly appropriate to investigate semantic compositionality in these models, though this has been studied comparatively little. There are many well-known mathematical operators on vectors that may be used for these purposes, many of which are well-known and heavily used in quantum mechanics.

We have summarized many of these operators, partly in the hope that gathering these descriptions into a single paper will enable researchers to more readily compare the different options and design experiments. We have performed some initial experiments that show that in at least some cases, composition operators can enable better modelling of similarities and relationships. We believe that this field is ripe for further development, and that contributing to a shared toolkit for researchers will expedite this development. To these ends, we have released tools for creating and exploring WORDSPACE models through the Semantic Vectors project, which is hosted at `http://code.google.com/p/semanticvectors` (Widdows & Ferraro 2008).

References

Aerts, D., and Czachor, M. 2004. Quantum aspects of semantic analysis and symbolic artificial intelligence. *J. Phys. A: Math. Gen.* 37:L123–L132.

Baeza-Yates, R., and Ribiero-Neto, B. 1999. *Modern Information Retrieval*. Addison Wesley / ACM Press.

Birkhoff, G., and von Neumann, J. 1936. The logic of quantum mechanics. *Annals of Mathematics* 37:823–843.

Boole, G. 1854. *An Investigation of the Laws of Thought*. Macmillan. Dover edition, 1958.

Bruza, P., and Cole, R. J. 2005. Quantum logic of semantic space: An exploratory investigation of context effects in practical reasoning. In Artëmov, S. N.; Barringer, H.; d'Avila Garcez, A. S.; Lamb, L. C.; and Woods, J., eds., *We Will Show Them! (1)*, 339–362. College Publications.

Bruza, P.; Lawless, W.; van Rijsbergen, K.; and Sofge, D., eds. 2007. *Quantum Interaction: Papers from the AAAI Spring Symposium*. Stanford, California: AAAI.

Clark, S., and Pulman, S. 2007. Combining symbolic and distributional models of meaning. In Bruza et al. (2007), 52–55.

Cristianini, J. S.-T. . N. 2000. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

Deerwester, S.; Dumais, S.; Furnas, G.; Landauer, T.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41(6):391–407.

Fellbaum, C., ed. 1998. *WordNet: An Electronic Lexical Database*. Cambridge MA: MIT Press.

Frege, G. 1884. *The Foundations of Arithmetic (1884)*. Blackwell, translated by J. L. Austin, 1974 edition.

Fulton, W., and Harris, J. 1991. *Representation theory - a first course*. Number 129 in Graduate Texts in Mathematics. Springer-Verlag.

Table 2: Verb Noun Similarities for Several Word Pairs

First Pair		Second Pair		Vector Sum	Direct Sum (Average Sim)	Tensor Product (Product Sim)
earn_vvi	money_nn1	pay_vvi	wages_nn2	0.66	0.41	0.16
earn_vvi	money_nn1	wages_nn2	pay_vvi	0.66	0.51	0.26
earn_vvi	money_nn1	pay_vvi	apple_nn1	0.40	0.25	0.01
earn_vvi	money_nn1	apple_nn1	pay_vvi	0.40	0.24	-0.02
eat_vvi	apple_nn1	eat_vvi	tomato_nn1	0.73	0.65	0.29
eat_vvi	apple_nn1	cook_vvi	tomato_nn1	0.52	0.44	0.17
eat_vvi	apple_nn1	throw_vvi	apple_nn1	0.57	0.52	0.04
eat_vvi	apple_nn1	throw_vvi	tomato_nn1	0.29	0.16	0.01

Gärdenfors, P. 2000. *Conceptual Spaces: The Geometry of Thought*. Bradford Books MIT Press.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Jänich, K. 1994. *Linear algebra*. Undergraduate Texts in Mathematics. Springer-Verlag.

Jurafsky, D., and Martin, J. H. 2000. *Speech and Language Processing*. New Jersey: Prentice Hall.

Kanerva, P. 1988. *Sparse Distributed Memory*. MIT Press.

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

Leech, G.; Garside, R.; and Bryant, M. 1994. Claws4: The tagging of the british national corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, 622–628.

Lund, K., and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments and computers* 28(22):203–208.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Partee, B. H.; ter Meulen, A.; and Wall, R. E. 1993. *Mathematical Methods in Linguistics*. Kluwer.

Plate, T. 2003. *Holographic Reduced Representations: Distributed Representation for Cognitive Structures*. CSLI Publications.

Pustejovsky, J. 1995. *The Generative Lexicon*. Cambridge, MA: MIT press.

Rieffel, E. 2007. Certainty and uncertainly in quantum information processing. In Bruza et al. (2007), 134–141.

Sahlgren, M. 2005. An introduction to random indexing. In *Proceedings of the Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering (TKE)*. Copenhagen, Denmark: SICS, Swedish Institute of Computer Science.

Sahlgren, M. 2006. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. Dissertation, Department of Linguistics, Stockholm University.

Salton, G., and McGill, M. 1983. *Introduction to modern information retrieval*. New York, NY: McGraw-Hill.

Salton, G.; Fox, E. A.; and Wu, H. 1983. Extended boolean information retrieval. *Communications of the ACM* 26(11):1022–1036.

Schütze, H. 1998. Automatic word sense discrimination. *Computational Linguistics* 24(1):97–124.

Smadja, F. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics* 19(1):143–177.

Smolensky, P. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46(1-2):159–216.

Szabó, Z. G. 2007. Compositionality. In Zalta, E. N., ed., *The Stanford Encyclopedia of Philosophy*.

van Rijsbergen, C. 1986. A non-classical logic for information retrieval. *The Computer Journal* 29:481–485.

van Rijsbergen, C. 2004. *The Geometry of Information Retrieval*. Cambridge University Press.

Varadarajan, V. S. 1985. *Geometry of Quantum Theory*. Springer-Verlag.

Widdows, D., and Bruza, P. 2007. Quantum information dynamics and open world science. In Bruza et al. (2007), 126–133.

Widdows, D., and Ferraro, K. 2008. Semantic vectors: A scalable open source package and online technology management application. In *Proceedings of the sixth international conference on Language Resources and Evaluation (LREC 2008)*.

Widdows, D. 2003. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Widdows, D. 2004. *Geometry and Meaning*. Stanford, California: CSLI publications.