Semantic Composition with Quotient Algebras

Daoud Clarke
University of Hertfordshire
Hatfield, UK
daoud@metrica.net

Rudi Lutz
University of Sussex
Brighton, UK
rudil@sussex.ac.uk

David Weir
University of Sussex
Brighton, UK
davidw@sussex.ac.uk

Abstract

We describe an algebraic approach for computing with vector based semantics. The tensor product has been proposed as a method of composition, but has the undesirable property that strings of different length are incomparable. We consider how a quotient algebra of the tensor algebra can allow such comparisons to be made, offering the possibility of data-driven models of semantic composition.

1 Introduction

Vector based techniques have been exploited in a wide array of natural language processing applications (Schütze, 1998; McCarthy et al., 2004; Grefenstette, 1994; Lin, 1998; Bellegarda, 2000; Choi et al., 2001). Techniques such as latent semantic analysis and distributional similarity analyse contexts in which terms occur, building up a vector of features which incorporate aspects of the meaning of the term. This idea has its origins in the distributional hypothesis of Harris (1968), that words with similar meanings will occur in similar contexts, and vice-versa.

However, there has been limited attention paid to extending this idea beyond individual words, so that the distributional meaning of phrases and whole sentences can be represented as vectors. While these techniques work well at the word level, for longer strings, data becomes extremely sparse. This has led to various proposals exploring methods for *composing* vectors, rather than deriving them directly from the data (Landauer and Dumais, 1997; Foltz et al., 1998; Kintsch, 2001; Widdows, 2008; Clark et al., 2008; Mitchell and Lapata, 2008; Erk and Pado, 2009; Preller and Sadrzadeh, 2009). Many of these approaches use a pre-defined composition operation such as addition (Landauer and Dumais, 1997; Foltz et al., 1998) or the tensor product (Smolensky, 1990; Clark and Pulman, 2007; Widdows, 2008) which contrasts with the *data-driven* definition of composition developed here.

2 Tensor Algebras

Following the context-theoretic semantics of Clarke (2007), we take the meaning of strings as being described by a multiplication on a vector space that is bilinear with respect to the addition of the vector space, i.e.

$$x(y+z) = xy + xz \quad (x+y)z = xz + yz$$

It is assumed that the multiplication is associative, but *not* commutative. The resulting structure is an **associative algebra over a field** — or simply an **algebra** when there is no ambiguity.

One commonly used bilinear multiplication operator on vector spaces is the **tensor product** (denoted $\otimes$), whose use as a method of combining meaning was first proposed by Smolensky (1990), and has been considered more recently by Clark and Pulman (2007) and Widdows (2008), who also looked at the **direct sum** (which Widdows calls the direct product, denoted $\oplus$).

We give a very brief account of the tensor product and direct sum in the finite-dimensional case; see (Halmos, 1974) for formal and complete definitions. Roughly speaking, if $u_1, u_2, \ldots u_n$ form an orthonormal basis for a vector space U and $v_1, v_2, \ldots v_m$ form an orthonormal basis for vector space V, then the space $U \otimes V$ has dimensionality nm with an orthonormal basis formed by the set of all ordered pairs (u_i, v_j), denoted by $u_i \otimes v_j$, of the individual basis elements. For arbitrary elements $u = \sum_{i=1}^{n} \alpha_i u_i$ and $v = \sum_{j=1}^{m} \beta_j v_j$ the tensor product of u and v is then given by

$$u \otimes v = \sum_{i}^{n} \sum_{j}^{m} \alpha_i \beta_j \; u_i \otimes v_j$$

For two finite dimensional vector spaces U and V (over a field F) of dimensionality n and m respectively, the direct sum $U \oplus V$ is defined as the cartesian product $U \times V$ together with the operations $(u_1, v_1) + (u_2, v_2) = (u_1 + u_2, v_1 + v_2)$, and $a(u_1, v_1) = (au_1, av_1)$, for $u_1, u_2 \in U$, $v_1, v_2 \in V$ and $a \in F$. In this case the vectors $u_1, u_2, \ldots u_n, v_1, v_2, \ldots v_m$ form an orthonormal set of basis vectors in $U \oplus V$, which is thus of dimensionality $n + m$. In this case one normally identifies U with the set of vectors in $U \oplus V$ of the form $(u, 0)$, and V with the set of vectors of the form $(0, v)$. This construction makes $U \oplus V$ isomorphic to $V \oplus U$, and thus the direct sum is often treated as commutative, as we do in this paper.

The motivation behind using the tensor product to combine meanings is that it is very fine-grained. So, if, for example, *red* is represented by a vector u consisting of a feature for each noun that is modified by *red*, and *apple* is represented by a vector v consisting of a feature for each verb that occurs with *apple* as a direct object, then *red apple* will be represented by $u \otimes v$ with a non-zero component for every pair of non-zero features (one from u and one from v). So, there is a non-zero element for each composite feature, *something that has been described as red*, and *something that has been done with an apple*, for example, *sky* and *eat*.

Both $\oplus$ and $\otimes$ are intuitively appealing as semantic composition operators, since u and v are reconstructible from each of $u \otimes v$ and $u \oplus v$, and thus no information is lost in composing u and v. Conversely, this is not possible with ordinary vector addition, which also suffers from the fact that it is strictly commutative (not simply up to isomorphism like $\oplus$), whereas natural language composition is in general manifestly non-commutative.

We make use of a construction called the **tensor algebra** on a vector space V (where V is a space of context features), defined as:

$$T(V) = \mathbb{R} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots$$

Any element of $T(V)$ can be described as a sum of components with each in a different tensor power of V. Multiplication is defined as the tensor product on these components, and extended linearly to the whole of $T(V)$. We define the **degree** of a vector u in $T(V)$ to be the tensor power of its highest dimensional non-zero component, and denote it $\deg(u)$; so for example, both $v \otimes v$ and $u \oplus (v \otimes v)$ have degree two, for $0 \neq u, v \in V$. We restrict $T(V)$ to only contain vectors of finite degree.

A standard way to compare elements of a vector space is to make use of an **inner product**, which provides a measure of semantic distance on that space. Assuming we have an inner product $\langle \cdot, \cdot \rangle$ on V, $T(V)$ can be given an inner product by defining $\langle \alpha, \beta \rangle = \alpha\beta$ for $\alpha, \beta \in \mathbb{R}$, and

$$\langle x_1 \otimes y_1, x_2 \otimes y_2 \rangle = \langle x_1, x_2 \rangle \langle y_1, y_2 \rangle$$

for $x_1, y_1, x_2, y_2 \in V$, and then extending this inductively (and by linearity) to the whole of $T(V)$.

We assume that words are associated with vectors in V, and that the higher tensor powers represent strings of words. The problem with the tensor product as a method of composition, given the inner product as we have defined it, is that strings of different lengths will have orthogonal vectors, clearly a serious problem, since strings of different lengths can have similar meanings. In our previous example, the vector corresponding to the concept *red apple* lives in the vector space $U \otimes V$, and so we have no way to compare it to the space V of nouns, even though *red apple* should clearly be related to *apple*.

Previous work has not made full use of the tensor product space; only tensor products are used, not sums of tensor products, giving us the equivalent of the **product states** of quantum mechanics. Our approach imposes relations on the vectors of the tensor product space that causes some product states to become equivalent to **entangled states**, containing sums of tensor products of different degrees. This allows strings of different lengths to share components. We achieve this by constructing a **quotient algebra**.

3 Quotient Algebras

An **ideal** I of an algebra A is a sub-vector space of A such that $xa \in I$ and $ax \in I$ for all $a \in A$ and all $x \in I$. An ideal introduces a congruence $\equiv$ on A defined by $x \equiv y$ if and only if $x - y \in I$. For any set of elements $\Lambda \subseteq A$ there is a unique minimal ideal I_Λ containing all elements of Λ; this is called the ideal **generated** by Λ. The quotient algebra A/I is the set of all equivalence classes defined by this congruence. Multiplication is defined on A/I by the multiplication on A, since $\equiv$ is a congruence.

By adding an element $x - y$ to the generating set Λ of an ideal, we are saying that we want to set $x - y$ to zero in the quotient algebra, which has the effect of setting x equal to y. Thus, if we

have a set of pairs of vectors that we wish to make equal in the quotient algebra, we put their differences in the generating set of the ideal. Note that putting a single vector v in the generating set can have knock-on effects, since all products of v with elements of A will also end up in the ideal.

Although we have an inner product defined on $T(V)$, we are not aware of any satisfactory method for defining an inner product on $T(V)/I$, a consequence of the fact that both $T(V)$ and I are not complete. Instead, we define an inner product on a space which contains the quotient algebra, $T(V)/I$. Rather than considering all elements of the ideal when computing the quotient, we consider a sub-vector space of the ideal, limiting ourselves to the space G_k generated from Λ by only allowing multiplication by elements up to a certain degree, k.

Let us denote the vector subspace generated by linearity alone (no multiplications) from a subset Λ of $T(V)$ by $G(\Lambda)$. Also suppose $B = \{e_1, \ldots, e_N\}$ is a basis for V. We then define the spaces G_k as follows. Define sets Λ_k ($k = 0, 1, 2, \ldots$) inductively as follows:

$$\Lambda_0 = \Lambda$$
$$\Lambda_k = \Lambda_{k-1} \cup \{(e_i \otimes \Lambda_{k-1}) | e_i \in B\} \cup \{(\Lambda_{k-1} \otimes e_i) | e_i \in B\}$$

Define

$$G_k = G(\Lambda_k)$$

We note that

$$G_0 \subseteq G_1 \subseteq \ldots G_k \subseteq \ldots \subseteq I \subseteq T(V)$$

form an increasing sequence of linear vector subspaces of $T(V)$, and that

$$I = \bigcup_{k=0}^{\infty} G_k$$

This means that for any $x \in I$ there exists a smallest k such that for all $k' \geq k$ we have that $x \in G_{k'}$.

Lemma. *Let $x \in I, x \neq 0$ and let $\deg(x) = d$. Then for all $k \geq d - mindeg(\Lambda)$ we have that $x \in G_k$, where $mindeg(\Lambda)$ is defined to be the minimum degree of the non-zero components occurring in the elements of Λ.*

Proof. We first note that for $x \in I$ it must be the case that $\deg(x) \geq mindeg(\Lambda)$ since I is generated from Λ. Therefore we know $d - mindeg(\Lambda) \geq 0$. We only need to show that $x \in G_{d-mindeg(\Lambda)}$. Let k' be the smallest integer such that $x \in G_{k'}$. Since $x \notin G_{k'-1}$ it must be the case that the highest degree term of x comes from $V \otimes G_{k'-1} \cup G_{k'-1} \otimes V$. Therefore $k' + mindeg(\Lambda) \leq d \leq k' + maxdeg(\Lambda)$. From this it follows that the smallest k' for which $x \in G_{k'}$ satisfies $k' \leq d - mindeg(\Lambda)$, and we know $x \in G_k$ for all $k \geq k'$. In particular $x \in G_k$ for $k \geq d - mindeg(\Lambda)$. □

We show that $T(V)/G_k$ (for an appropriate choice of k) captures the essential features of $T(V)/I$ in terms of equivalence:

Proposition. *Let $\deg(a - b) = d$ and let $k \geq d - mindeg(\Lambda)$. Then $a \equiv b$ in $T(V)/G_k$ if and only if $a \equiv b$ in $T(V)/I$.*

Proof. Since $G_k \subseteq I$, the equivalence class of an element a in $T(V)/I$ is a superset of the equivalence class of a in $T(V)/G_k$, which gives the forward implication. The reverse follows from the lemma above. □

In order to define an inner product on $T(V)/G_k$, we make use of the result of Berberian (1961) that if M is a finite-dimensional linear subspace of a pre-Hilbert space P, then $P = M \oplus M^{\perp}$, where $M^{\perp}$ is the orthogonal complement of M in P. In our case this implies $T(V) = G_k \oplus G_k^{\perp}$ and that every element $x \in T(V)$ has a unique decomposition as $x = y + x'_k$ where $y \in G_k$ and $x'_k \in G_k^{\perp}$. This implies that $T(V)/G_k$ is isomorphic to $G_k^{\perp}$, and that for each equivalence class $[x]_k$ in $T(V)/G_k$ there is a unique corresponding element $x'_k \in G_k^{\perp}$ such that $x'_k \in [x]_k$. This element x'_k can be thought of as the **canonical representation** of all elements of $[x]_k$ in $T(V)/G_k$, and can be found by projecting any element in an equivalence class onto $G_k^{\perp}$. This enables us to define an inner product on $T(V)/G_k$ by $\langle [x]_k, [y]_k \rangle_k = \langle x'_k, y'_k \rangle$.

The idea behind working in the quotient algebra $T(V)/I$ rather than in $T(V)$ is that the elements of the ideal capture differences that we wish to ignore, or alternatively, equivalences that we wish to impose. The equivalence classes in $T(V)/I$ represent this imposition, and the canonical representatives in $I^{\perp}$ are elements which ignore the distinctions between elements of the equivalence classes.

However, by using G_k, for some k, instead of the full ideal I, we do not capture some of the equivalences implied by I. We would, therefore, like to choose k so that no equivalences of importance *to the sentences we are considering* are ignored. While we have not precisely established a minimal value for k that achieves this, in the discussion that follows, we set k heuristically as

$$k = l - mindeg(\Lambda)$$

where l is the maximum length of the sentences currently under consideration, and Λ is the generating set for the ideal I. The intuition behind this is that we wish all vectors occurring in Λ to have some component in common with the vector representation of our sentences. Since components in the ideal are generated by multiplication (and linearity), in order to allow the elements of Λ containing the lowest degree components to potentially interact with our sentences, we will have to allow multiplication of those elements (and all others) by components of degree up to $l - mindeg(\Lambda)$.

Given a finite set $\Lambda \subseteq T(V)$ of elements generating the ideal I, to compute canonical representations, we first compute a generating set Λ_k for G_k following the inductive definition given earlier, and removing any elements that are not linearly independent using a standard algorithm. Using the Gram-Schmidt process (Trefethen and Bau, 1997), we then calculate an orthonormal basis Λ' for G_k, and, by a simple extension of Gram-Schmidt, compute the projection of a vector u onto $G_k^\perp$ using the basis Λ'.

We now show how Λ, the set of vectors generating the ideal, can be constructed on the basis of a tree-bank, ensuring that the vectors for any two strings of the same grammatical type are comparable.

4 Data-driven Composition

Suppose we have a tree-bank, its associated tree-bank grammar G, and a way of associating a context vector with every occurrence of a subtree in the tree-bank (where the vectors indicate the presence of features occurring in that particular context). The context vector associated with a specific occurrence of a subtree in the tree-bank is an **individual** context vector.

We assume that for every rule, there is a distinguished non-terminal on the right hand side which we call the head. We also assume that for every production π there is a linear function ϕ_π from the space generated by the individual context vectors of the head to the space generated by the individual context vectors of the left hand side. When there is no ambiguity, we simply denote this function ϕ.

Let $\widehat{X}$ be the sum over all individual vectors of subtrees rooted with X in the tree-bank. Similarly, for each X_j in the right-hand-side of the rule $\pi_i : X \to X_1 \dots X_{r(\pi_i)}$, where $r(\pi)$ is the rank of π, let $\widehat{\pi_{i,j}}$ be the sum over the individual vectors of those subtrees rooted with X_j where the subtree occurs as the jth daughter of a local tree involving the production π_i in the tree-bank.

For each rule $\pi : X \to X_1 \dots X_r$ with head X_h we add vectors

$$\lambda_{\pi,i} = \phi(e_i) - \widehat{X}_1 \otimes \dots \otimes \widehat{X}_{h-1} \otimes e_i \otimes \widehat{X}_{h+1} \otimes \dots \otimes \widehat{X}_r$$

for each basis element e_i of V_{X_h} to the generating set. The reasoning behind this is to ensure that the meaning corresponding to a vector associated with the head of a rule is maintained as it is mapped to the vector space associated with the left hand side of the rule.

It is often natural to assume that the individual context vector of a non-terminal is the same as the individual context vector of its head. In this case, we can take ϕ to be the identity map. In particular, for a rule of the form $\pi : X \to X_1$, then $\lambda_{\pi,i}$ is zero.

It is important to note at this point that we have presented only one of many ways in which a grammar could be used to generate an ideal. In particular, it is possible to add more vectors to the ideal, allowing more fine-grained distinctions, for example through the use of a lexicalised grammar.

For each sentence w, we compute the tensor product $\hat{w} = \hat{a}_1 \otimes \hat{a}_2 \otimes \dots \otimes \hat{a}_n$ where the string of words $a_1 \dots a_n$ form w, and each $\hat{a}_i$ is a vector in V. For a sentence w we find an element $\hat{w}_O$ of the orthogonal complement of G_k in $T(V)$ such that $\hat{w}_O \in [\hat{w}]$, where $[\hat{w}]$ denotes the equivalence class of $\hat{w}$ given the subspace G_k.

5 Example

We show how our formalism applies in a simple example. Assume we have a corpus which consists of the following sentences:

	apple	*big apple*	*red apple*	*city*	*big city*	*red city*	*book*	*big book*	*red book*
apple	1.0	0.26	0.24	0.52	0.13	0.12	0.33	0.086	0.080
big apple		1.0	0.33	0.13	0.52	0.17	0.086	0.33	0.11
red apple			1.0	0.12	0.17	0.52	0.080	0.11	0.33
city				1.0	0.26	0.24	0.0	0.0	0.0
big city					1.0	0.33	0.0	0.0	0.0
red city						1.0	0.0	0.0	0.0
book							1.0	0.26	0.24
big book								1.0	0.33
red book									1.0

Figure 1: Similarities between phrases

see red apple
buy apple
read big book
throw old small red book
buy large new book
see big city
visit big apple
modernise city
see modern city

together with the following productions.

1. $\mathrm{N}' \to \mathrm{Adj}\ \mathrm{N}'$
2. $\mathrm{N}' \to \mathrm{N}$

where N and Adj are terminals representing nouns and adjectives, along with rules for the terminals. We consider the space of adjective/noun phrases, generated by N', and define the individual context of a noun to be the verb it occurs with, and the individual context of an adjective to be the noun it modifies. For each rule, we take ϕ to be the identity map, so the vector spaces associated with N and N', and the vector space generated by individual contexts of the nouns are all the same. In this case, the only non-zero vectors which we add to the ideal are those for the second rule (ignoring the first rule, since we do not consider verbs in this example except as contexts), which has the set of vectors

$$\lambda_i = e_i - \widehat{\mathrm{Adj}} \otimes e_i$$

where i ranges over the basis vectors for contexts of nouns: see, buy, $visit$, $read$, $modernise$, and

$$\widehat{\mathrm{Adj}} = 2e_{apple} + 6e_{book} + 2e_{city}$$

In order to compute canonical representations of vectors, we take $k = 1$.

5.1 Discussion

Figure 1 shows the similarity between the noun phrases in our sample corpus. Note that the vectors we have put in the generating set describe only compositionality of meaning — thus for example the similarity of the non-compositional phrase *big apple* to *city* is purely due to the distributional similarity between *apple* and *city* and composition with the adjective *big*.

Our preliminary investigations indicate that the cosine similarity values are very sensitive to the particular corpus and features chosen; we are currently investigating other ways of measuring and computing similarity.

One interesting feature in the results is how adjectives alter the similarity between nouns. For example, *red apple* and *red city* have the same similarity as *apple* and *city*, which is what we would expect from a pure tensor product. This also explains why all phrases containing *book* are disjoint to those containing *city*, since the original vector for *book* is disjoint to *city*.

The contribution that the quotient algebra gives is in comparing the vectors for nouns with those for noun-adjective phrases. For example, *red apple* has components in common with *apple*, as we would expect, which would not be the case with just the tensor product.

6 Conclusion and Further Work

We have presented the outline of a novel approach to semantic composition that uses quotient algebras to compare vector representations of strings of different lengths.

The dimensionality of the construction we use increases exponentially in the length of the sentence; this is a result of our use of the tensor product. This causes a problem for computation using longer phrases; we hope to address this in future work by looking at the representations we use. For example, product states can be represented in much lower dimensions by representing them as products of lower dimensional vectors.

The example we have given would seem to indicate that we intend putting abstract (syntactic) information about meaning into the set of generating elements of the ideal. However, there is no reason that more fine-grained aspects of meaning cannot be incorporated, even to the extent of putting in vectors for every pair of words. This would automatically incorporate information about non-compositionality of meaning. For example, by including the vector $\widehat{big\ apple} - \widehat{big} \otimes \widehat{apple}$, we would expect to capture the fact that the term *big apple* is non-compositional, and more similar to *city* than we would otherwise expect.

Future work will also include establishing the implications of varying the constant k and exploring different methods for choosing the set Λ that generates the ideal. We are currently preparing an experimental evaluation of our approach, using vectors obtained from large corpora.

7 Acknowledgments

We are grateful to Peter Hines, Stephen Clark, Peter Lane and Paul Hender for useful discussions. The first author also wishes to thank Metrica for supporting this research.

References

Jerome R. Bellegarda. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE*, 88(8):1279–1296.

Sterling K. Berberian. 1961. *Introduction to Hilbert Space*. Oxford University Press.

Freddy Choi, Peter Wiemer-Hastings, and Johanna Moore. 2001. Latent Semantic Analysis for text segmentation. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 109–117.

Stephen Clark and Stephen Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 52–55, Stanford, CA.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pages 133–140, Oxford, UK.

Daoud Clarke. 2007. *Context-theoretic Semantics for Natural Language: an Algebraic Framework*. Ph.D. thesis, Department of Informatics, University of Sussex.

Katrin Erk and Sebastian Pado. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the EACL Workshop on Geometrical Methods for Natural Language Semantics (GEMS)*.

P. W. Foltz, W. Kintsch, and T. K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Process*, 15:285–307.

Gregory Grefenstette. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers, Dordrecht, NL.

Paul Halmos. 1974. *Finite dimensional vector spaces*. Springer.

Zellig Harris. 1968. *Mathematical Structures of Language*. Wiley, New York.

W. Kintsch. 2001. Predication. *Cognitive Science*, 25:173–202.

T. K. Landauer and S. T. Dumais. 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.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, pages 768–774, Montreal.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 279, Morristown, NJ, USA. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

Anne Preller and Mehrnoosh Sadrzadeh. 2009. Bell states and negation in natural languages. In *Proceedings of Quantum Physics and Logic*.

Heinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

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

Lloyd N. Trefethen and David Bau. 1997. *Numerical Linear Algebra*. SIAM.

Dominic Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second Symposium on Quantum Interaction, Oxford, UK*.