Distributional Compositional Semantics

Kendra Plante

1 Introduction

We begin this paper by describing two different approaches to semantics. One the hand, there is the traditional approach to semantics, based on the philiosophy of Gottlob Frege. This approach is *compositional* and *truth-functional*. Respectively, these notions express that the meanings of individual words can be combined, or composed, to form the meanings of strings, and that the meaning of a sentence is its truth value. The basic assumption underlying this philosophy is that the meaning of a string is a function of the meanings of its parts.

For example, one model of compositional semantics is based on type theory. The meaning of a noun is its extension, namely the set of objects in the universe corresponding to it, or the characteristic function of this set. The meaning of an intransitive verb is then a function from nouns (characteristic functions) to truth values (the discrete set $\{0, 1\}$), for transitive verbs, a function from two nouns to truth values, etc. We express these definitions in the following notation: Let e denote the type of a noun and t the type of a truth value. Then the types of nouns and verbs are as follows:

1. Noun: e
2. Intransitive verb: $e \to t$
3. Transitive verb: $e \to (e \to t)$

The type of the transitive verb given above is justified by an adjunction known as currying, namely

$$\mathrm{Hom}(A \times B, C) \cong \mathrm{Hom}(A, \mathrm{Hom}(B, C))$$

which expresses the fact that a set function of two variables can be thought of instead as a function from one set to a set of functions between sets, without any loss of information. This adjunction will be useful to us later as well.

One clear limitation of this model of semantics is that the meaning of a given sentence, or even of a word, is very tightly constrained. For example, the sentences "The sky is blue" and "Clouds are white" are both true, hence, according to the above model, they have the same meaning. But this is not completely satisfactory, as there seems to be a sense in which they have distinct meanings, namely in the fact that they express different thoughts. Furthermore,

the sentence "$2 + 2 = 4$" is also true, hence has the same meaning as the above two, but there is also a sense in which we want to say that the first two are more *similar* in meaning than the third, in that they both express facts about color, or objects related to the sky, etc.

The same sort of problem arises on the level of individual words as well. Consider, for example, the famous issue regarding the nouns "cordate" and "renate". Respectively, these refer to creatures with hearts and creatures with kidneys. It is a fact of biology that all creatures with hearts also have kidneys, and vice versa, i.e. a creature is a cordate if and only if it is a renate. Consequently, these words have the same extension. It follows from the above semantic model that they have the same meaning. Once again, this is unsatisfactory, because they clearly express different thoughts. In the language of Frege, one may say they have different *intensions*. We want a more versatile semantics which can account for this.

A different approach to semantics which attempts to remedy these concerns is the distributional approach. This approach is based loosely on Wittgenstein's philosophy of "meaning in use", i.e. that we can understand the meaning of a word by investigating how it is used in real discourse in relation to other words. The term *distributional* here refers to a model which describes the meaning of a word according to its usage relative to other words in some corpus.

One example of a model which takes this approach to semantics is based on vector spaces. The basic idea is to choose some set of lemmas B (say, nouns, for simplicity) from a corpus and generate a vector space V with B as a basis. The basis elements B are referred to as *context words*. We then describe a process for representing any other word in the corpus as a weighted vector sum of these basis elements. Typically, this process involves counting the occurrences of each context word in sentences containing the word with some coefficients (called *weights*) to correct for the relative importance of certain contexts over others. This procedure will generate a vector for each word in the corpus, and we define the *meaning* of the word to be this vector.

This approach has the immediate advantage that we can compare meanings of words directly, regardless of their extension. The idea is that words more similar in meaning will result in vectors which are nearer to each other in the vector space. We can quantify this by the cosine of the angle between them (or equivalently, the normalized inner product). The smaller the value, the more similar the meaning.

For example, consider a vector space which contains "food", "pet", and "police" as basis vectors. We would expect that the vector for the word "cat" would appear somewhere between "food" and "pet", and farther from "police", since we would expect to find mostly sentences such as "He has a cat as a pet" and "Get some cat food", whereas "crime" might appear closest to "police", because of sentences such as "The police investigated the crime". Finally, we can expect that "dog" would appear close to "cat", but perhaps also closer to "crime" than "cat", because of sentences like "There were police dogs at the crime scene". Combining all these expectations, we would expect to conclude from such a model that "cat" and "dog" are more similar in meaning than either

are to "crime".

We can also apply this model to the cordate-renate issue in a very elegant way: Take as basis vectors the words "heart" and "kidney". We would clearly expect that from any corpus, the vector corresponding to "cordate" would appear closest to "heart" and likewise for "renate" and "liver". Hence, we see the aforementioned concerns resolved exactly in accordance with our intuition.

One obvious limitation of this model as described is that it provides no way of combining words into strings, i.e. it is not compositional. The unfortunate consequence of this is that we are unable to say anything about the meaning of phrases or sentences. In this paper, we outline a model which combines the distributional and compositional approaches to semantics in order to reap the benefits of both.

2 Mathematical Background

In order to motivate and develop the distributional compositional model of semantics, it is necessary to build up some mathematical background. In particular, we will need to understand the notion of a compact closed monoidal category, and in particular, we will need to understand the natural compact closed monoidal structures on the category of finite dimensional vector spaces and the category of pregroups. Since we will only work with monoidal categories in this paper, we will introduce it as a single concept.

Definition 1. *A monoidal category consists of the following data:*

1. *A family C of objects*
2. *A set of morphisms $\mathrm{Hom}(A, B)$ for each object $A, B \in C$*
3. *An associative binary operation $\otimes : C \times C \to C$*

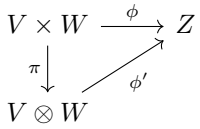
subject to the following axioms:

1. *For each ordered triple of objects (A, B, C) and each pair of morphisms $f : A \to B$ and $g : B \to C$, there is a composite morphism $g \circ f : A \to C$, where the composition $\circ : \mathrm{Hom}(B, C) \times \mathrm{Hom}(A, B) \to \mathrm{Hom}(A, C)$ is associative.*
2. *For each object A, there is an identity morphism $1_A : A \to A$ such that, for $f : A \to B$, we have $f \circ 1_A = f$ and $1_B \circ f = f$.*
3. *There is a unit object I which satisfies $I \otimes A = A = A \otimes I$.*
4. *For each ordered pair of morphisms $(f : A \to C, g : B \to D)$, there is a bifunctorial parallel composite morphism $f \otimes g : A \otimes B \to C \otimes D$, such that $(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2)$.*

We will typically use C alone to denote the category. Note that in this definition, the binary operation $\otimes$, subject to the axioms which govern it, is the *monoidal structure* on C.

Loosely speaking, a monoidal category is a category together with a binary operation on its objects which "acts" like multiplication. This operation is typically called the *tensor product* and denoted by the symbol $\otimes$. The reason for this is that the standard tensor product on vector spaces turns the category $\mathbf{Vect_K}$ of vector spaces (over a field K) into a monoidal category, where the identity 1 is given by the underlying field K. (In this paper, we will be concerned only with finite dimensional vector spaces over $\mathbb{R}$, which we will denote by **FVect**.)

Recall that the tensor product on vector spaces has the following additional structure: It is commutative up to isomorphisms, i.e. $V \otimes W \cong W \otimes V$. Also, every bilinear function factors uniquely through the tensor product, i.e. for every bilinear function $\phi : V \times W \to Z$, there is a unique linear function $\phi' : V \otimes W \to Z$ such that $\phi'(v \otimes w) = \phi(v, w)$. In other words, ϕ' is the unique map which makes the following diagram commute



where $\pi : V \times W \to V \otimes W$ is the canonical map $(v, w) \mapsto v \otimes w$. This is known as the *universal property* of the tensor product.

Finally, we have the following natural isomorphism known as the *tensor-hom adjunction* which generalizes the process of currying mentioned above

$$\mathrm{Hom}(X \otimes Y, Z) \cong \mathrm{Hom}(X, \mathrm{Hom}(Y, Z))$$

Another monoidal category which will be important to us is that of a pregroup.

Definition 2. *A pregroup is a set P together with a partial order relation $\leq$ and a monoidal multiplication $*$, such that for every $x \in P$, there exist elements $x^l, x^r \in P$ such that*

$$x^l * x \leq 1 \qquad x * x^r \leq 1$$
$$1 \leq x * x^l \qquad 1 \leq x^r * x$$

In practice, we will omit the symbol $*$ between elements and write ab instead of $a * b$. We let **Preg** denote the category of pregoups, with morphisms given by order preserving maps which commute with the monoidal multiplication.

The elements x^l and x^r are called the *left* and *right adjoints* of x, respectively. Note that every pregroup is a monoidal category itself with elements of

P as objects, a morphism $a \to b$ whenever $a \leq b$, and $*$ as the monoidal multiplication. Note that this multiplication is in general not commutative. The additional structure present in this category, namely that characterized by the left and right adjoints, is a special case of a property called *compact closure*.

Definition 3. *A monoidal category is compact closed if for each object A there are objects A^l and A^r such that there exist maps*

$$A^l \otimes A \to 1 \qquad A \otimes A^r \to 1$$
$$1 \to A \otimes A^l \qquad 1 \to A^r \otimes A$$

Observe that the category of finite dimensional vector spaces with tensor product is also compact closed, where the left and right adjoint of any vector space is its dual, i.e. $V^l = V^r = V^*$. Finally, since V is finite dimensional, $V^* = V$. Hence, each element is its own left and right adjoint. Thus, in **FVect**, the maps $A^l \otimes A \to 1$ and $A \otimes A^r \to 1$ are both the same map $A \otimes A \to \mathbb{R}$. It can be shown that this map is given explicitly by

$$\sum_{i,j} c_{ij} v_i \otimes w_j \mapsto \sum_{i,j} c_{ij} \langle v_i, w_j \rangle$$

where $\langle , \rangle$ is the standard inner product. This makes it computable, which will be important to us later.

The key idea of this paper will be exploiting this common structure between pregroups and finite dimensional vector spaces with tensor product in order to construct a semantics which is both compositional and distributional. But before we begin this construction, we must outline how pregroups can be used to model syntax.

3 Lambek Calculus

In order to model syntax using the algebra of pregroups, we will assign grammatical types to each word, use the monoidal multiplication to combine them, and use the adjoint relations to reduce complex types to phrase or sentence types. The particular schema we will describe is known as the *Lambek calculus*. Since we only need a very basic grammar for the purposes of this paper, we will stick to basic sentences of the form **noun-verb-noun**.

Let n denote the type of a noun and s the type of a (grammatical) sentence. Then a grammar which contains nouns, intransitive, and transitive verbs has the following types:

1. Noun: n
2. Intransitive verb: $n^r s$
3. Transitive verb: $n^r s n^l$

The use of the adjoints in the verb types express the fact that, for example, a transitive verb must take a noun on the left and on the right in order to form a grammatical sentence. This is best illustrated by an example.

Say we have the sentence "Jack likes Jill". Gluing together the types of each word in the proper order, we find that this sentence has the complex type $n(n^r sn^l)n$, with parentheses used only to indicate which type comes from which word. Using associativity of the multiplication and applying the properties which define the adjoints, we obtain the following reduction

$$n(n^r sn^l)n \to (nn^r)s(n^l n) \to 1s1 \to s$$

The interpretation of this computation is that "Jack likes Jill" has the type of a sentence, which is to say that it is grammatical.

4 Distributional Semantics

The general notion of distributional semantics, as well a common and practical model involving vector spaces, was explained in the introduction. We recall the basics here: Choose from some corpus X a finite set of lemmas $B = \{b_1, \ldots, b_n\}$ and let V be the vector space generated by B. For each word $x \in X$, take the set of sentences Y in X which contain x. For each $b_i \in B$, let a_i denote the number of times b_i occurs in Y, i.e. the number of times x and b_i occur in the same sentence. We then assign to x the vector in V given by

$$x = \sum_{i=1}^{n} a_i b_i$$

Note that we may also assign weights to the coefficients a_i to account for the importance of the lemma b_i relative to x, and this is often done in practice, but it will not be necessary for our purposes to explore this.

We repeat this process for each $x \in X$ and we call this vector the *meaning* of x. This proves useful in capturing the meaning of x relative to the meanings of other words in X, in the sense that words with more similar meanings result in vectors which are closer together. With this in mind, we quantify the similarity between words in terms of the angle between them.

Definition 4. *For two words $x, y \in V$, we define the degree of similarity d between x and y to be*

$$d = \frac{\langle x, y \rangle}{|x||y|} = \cos\theta$$

where $\langle ., . \rangle$ is the usual inner product, $|.|$ is the usual norm, and θ is the angle between x and y.

Note that the above simply gives the definition in two different but equivalent ways: the cosine of the angle between them is easier to visualize, whereas the

normalized inner product provides a means of computing the degree without needing to know the angle explicitly before-hand.

These vector space models of distributional semantics have proved very effective in recent history at a variety of natural language processing tasks, such as automatic thesaurus building. It is therefore desirable to extend these models so that they are able to capture the semantics of composite linguistic expressions, such as phrases and sentences. In other words, given a pair of vectors x and y representing the meanings of two words in a corpus, can we construct a third vector z representing the meaning of the phrase xy?

The naive approach is to simply use a binary operation which is natural to vector spaces, such as vector addition, componentwise multiplication, or tensor product. These approaches each produce somewhat different theories, e.g. addition and componentwise multiplication can only be applied to vectors in the same ambient vector space, whereas tensor product can combine vectors from any two vector spaces and produce a vector in a third, often distinct vector space.

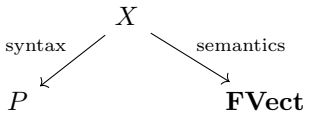
However, all of these fail to meet even the basic requirements of compositionality in natural languages, in that they are all commutative, whereas natural language is highly non-commutative. That is to say, $x+y = y+x$, but "Jack likes Jill" $\neq$ "Jill likes Jack". It is therefore clear that we need a non-commutative operation to accurately represent compositionality.

Unfortunately, there is no natural non-commutative binary operation on vector spaces. Some authors have attempted to construct some for this exact purpose, but these constructions have been complicated and ad hoc. However, if we combine the structure of vector space distributional semantics with the Lambek calculus, effectively accounting for the semantics and syntax in the same model, we will see that a solution to this problem arises very naturally. This is the task we will take up in the next section.

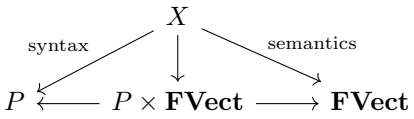
5 Combining the Models

Recall that pregroups and finite dimensional vector spaces with tensor product, as monoidal categories, share a common structure known as compact closure. In the case of pregroups and the Lambek calculus, this structure is essential to understanding compositionality, by allowing us a means of reducing complex grammatical types to simpler ones. The fact that these categories have this structure in common allows us, loosely speaking, to combine them into a single model which maintains the important aspects of each. Explicitly, this combination proceeds as follows:

Say we are given a corpus X. We can imagine the vector space and pregroup models for distributional semantics and syntax described above as two different ways of modeling the corpus, each being used to capture certain salient aspects of it in a way which allows us to make computations and obtain additional information which would not be possible from the corpus alone. We visualize this with a diagram



for some pregroup P. But we can combine them by forming the product $P \times \mathbf{FVect}$ with natural projections onto each component, i.e.



Since for each pregroup P, both P and **FVect** are monoidal and compact closed, it can be shown that these lift to a natural monoidal, compact closed structure on $P \times \mathbf{FVect}$. That is to say, there is a natural way that $P \times \mathbf{FVect}$ can be made into a compact closed monoidal category, such that, restricted to each component, these structures agree with the original structures.

For example, the monoidal structure on $P \times \mathbf{FVect}$ becomes the product $(a, V) * (b, W) = (ab, V \otimes W)$, i.e. simply applying the products in P and **FVect** componentwise. With this definition, each object $(a, V) \in P \times \mathbf{FVect}$ consists of a vector space V together with a grammatical type a, and elements of this object are vectors in V together with a grammatical type, (a, v).

The product on $P \times \mathbf{FVect}$ induces a product on these elements, namely $(a, v) * (b, w) = (ab, v \otimes w)$, which lives in the space $(ab, V \otimes W)$. Recall that the product in P is not commutative, since it is built precisely to model syntax, hence the product described above is also not commutative, i.e. $(ab, v \otimes w) \neq (ba, w \otimes v)$, even though $v \otimes w = w \otimes v$. In linguistics terms, this expresses exactly what we want: For strings of words x and y, the composite string xy does not necessraily have the same meaning as yx. In particular, "Jack likes Jill" is semantically distinct from "Jill likes Jack".

This solution is elegant because of its intuitive naturally: The underlying principle is that the strings xy and yx are distinct in meaning precisely because xy and yx are grammatically distinct. We can therefore expect that the difference between the meanings of xy and yx is somehow contained in their grammatical distinction, in terms of the meanings of x and y individually. This suggests a connection between syntax and semantics which we might expect.

We now define explicitly the *meaning* of a string in this model. We will start with an example to motivate the general definition. Consider the case of the grammatical sentence "Jack likes Jill". Using the Lambek calculus, this sentence has the grammatical reduction $n(n^r s n^l)n \to s$, as detailed above. Recall that this is simply a morphism in the pregroup P. If we lift this reduction to $P \times \mathbf{FVect}$, we get a linear map between vector spaces which respects this reduction. Explicitly, we get the following:

We need to assign vector spaces to each word in the sentence. Say that all nouns in our corpus live in a vector space denoted N and grammatical sentences in a vector space denoted S. Then, in order to respect the grammatical reduction $n(n^r s n^l)n \to s$, transitive verbs such as "like" must live in the vector space $N \otimes S \otimes N$. (Recall that $N^l = N^r = N^* = N$.) Let $g : P \to P$ denote the map $n(r^r s n^l)n \to s$. Then, with this notation in place, g lifts to a map $f : P \times \mathbf{FVect} \to P \times \mathbf{FVect}$ given by

$$(n(n^r s n^l)n, N \otimes N \otimes S \otimes N \otimes N) \mapsto (s, S)$$

such that f restricted to P is g. Recall definition 3: the existence of such maps, in each component, is part of what it means for these categories to be compact closed. To each grammatical reduction map, such as g, there exists some map $g' : \mathbf{FVect} \to \mathbf{FVect}$ corresponding to it, based on this structure. We then simple take $f = (g, g')$.

Generalizing this procedure, for each grammatical reduction, we get a map on $P \times \mathbf{FVect}$ corresponding to this reduction. Thus, after applying these functions, all strings of the same grammatical type, post-reduction, end up in the same vector space, and can be compared accordingly. In particular, all grammatical sentences end up in the same vector space.

We now return to the example "Jack likes Jill" to see exactly how this works in practice. We start by using our existing distributional model to assign vectors to each word the sentence. The nouns are simple, following the exact procedure outlined earlier in the paper. We express each as linear combinations of basis elements $\{v_i\}$ in N:

$$\text{Jack} = \sum_i a_i v_i$$

$$\text{Jill} = \sum_j b_j v_j$$

These vectors are each paired with the grammatical type n, for noun. The verb "likes" lives in the vector space $N \otimes S \otimes N$, as explained above, hence has the form

$$\text{likes} = \sum_{i,j,k} c_{ijk} v_i \otimes s_j \otimes w_k$$

where $\{s_j\}$ is a basis for S. This vector is paired with the grammatical type $n^r s n^l$. To compute the vector in S for the sentence "Jack likes Jill", we use the procedure described above, applying to Jack $\otimes$ likes $\otimes$ Jill the function f induced by the reduction $n(n^r s n^l)n \to s$. Mathematically, this gives us

$$f(\text{Jack} \otimes \text{likes} \otimes \text{Jill}) = \sum_{i,j,k} c_{ijk} \langle \text{Jack}, v_i \rangle s_j \langle w_k, \text{Jill} \rangle$$
$$= \sum_j \left(\sum_{i,k} c_{ijk} \langle \text{Jack}, v_i \rangle \langle w_k, \text{Jill} \rangle \right) s_j$$

This follows from the explicit representation of the reduction maps in **FVect** described earlier. The important thing to take away from this is that these vectors are entirely computable from the vectors associated to each word in the string, exactly as we would want from a compositional model.

This process guarantees us a well-defined vector in S for every grammatical sentence, which is computable from the vectors assigned to each of its component words using any vector space distributional model, using the Lambek calculus. Likewise, all phrases of grammatical type x will appear in the same vector space X. We therefore are able to quantify similarity between phrases and sentences in exactly the same way as we did for individual words above, namely by the cosine of the angle between their associated vectors. Explicitly, we have the following definition, which mirrors definition 4:

Definition 5. *For two strings of words x, y of the same grammatical type, we define the degree of similarity d between x and y to be*

$$d = \frac{\langle x, y \rangle}{|x||y|} = \cos\theta$$

where $\langle ., . \rangle$ is the usual inner product, $|.|$ is the usual norm, and θ is the angle between x and y.

We stipulate that x and y have the same grammatical type because we cannot compute their inner product if they live in different vector spaces. This is acceptable, though, because we would generally only be interested in comparing the semantics of strings of the same grammatical type. There are, however, possible ways this model could be expanded to remove this destriction, if so desired, e.g. by embedding two distinct vector spaces in the same ambient space, e.g. their Cartesian product. But we will not explore these possibilities here.

6 Applications

It is clear that a working model of distributional compositional semantics could be very useful for various applications to natural language processing, such as the way in which existing distributional models have already been applied successfully to automatic thesaurus building. But it is conceivable that it may lead to even further applications in less expected ways. We conclude this section

with a brief description of one of these potential applications, and describe how this model might be used to gain quantitative information regarding the relation between words, phrases, and sentences.

As motivation, say we have a notion of the meanings of the words "love" and "hate". Intuitively, the meaning of "like" is somewhere between the two, and closer to "love". We may say, for example,

$$\text{like} = \frac{3}{4}\text{love} + \frac{1}{4}\text{hate}$$

We can then use basic vector arithmetic to work with the semantics of strings containing "like" knowing only information about the semantics of strings containing "like" and "love". Working backwards, say we have a corpus and an algorithm based on this model which produces vectors for all strings in the corpus. We can then compute the degree of similarity between "like" and each of the words "love" and "hate", respectively, using the above formula. This allows us to write

$$\text{like} = a * \text{love} + b * \text{hate}$$

for some $a, b \in [0, 1]$, and experimentally determine exactly what a and b should be for each corpus. This could in turn give us interesting information about linguistic cultural differences. In particular, we can expect that for different sets of corpus data, the degrees of similarity between words would also vary. If we choose corpus data specifically from separate cultures which we are interested in comparing linguistically, the extent to which this degree varies from corpus to corpus, as well as the way in which it varies, can help us draw quantitative conclusions about how word meanings differ across cultures.

7 Experimental Results

In another study, Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh constructed algorithms for two different semantic models based on the approach outlined in this paper and tested it alongside other algorithms for distributional semantics which have been constructed to estimate the similarity of meanings between sentences. Their study used a corpus of about 6 million sentences and $500,000$ unique lemmas from the British National Corpus.

The various models were used to construct vectors for a set of sentences of similar structure (e.g. "the system met the criterion" vs. "the system visited the criterion"). A group of native English speaking volunteers were instructed to rank the sentences according to which were more similar to which. These rankings were compared to those produced by the algorithms. In all the tests performed, the models based on the categorical approach in this paper outperformed the others. These results can be read in [2].

8 Connections to Traditional Compositional Semantics

We conclude this paper by providing some interesting interpretations of the model constructed in this paper which reveal underlying connections between it and traditional compositional semantics. First of all, consider the semantics of a transitive verb. In the introduction to this paper, we explained how in traditional compositional semantics, transitive verbs are assigned the type $e \to (e \to t)$, which expresses (via currying) that they take two nouns and produce a truth value. In our distributional semantics, we end up with vectors, not necessarily truth values, but it is easily seen that the same underlying principle applies.

As described in section 5, the transitive verbs are associated to vectors in $N \otimes S \otimes N$, where N and S are vector spaces assigned to nouns and grammatical sentences, respectively. According to the distributional model, these vectors are precisely the meanings of these verbs. By commutativity of the tensor product (up to isomorphisms), we have $N \otimes S \otimes N \cong N \otimes N \otimes S$. There is also a well-known isomorphism $V^* \otimes W \cong \mathrm{Hom}(V, W)$ which holds for any finite dimensional vector spaces V and W. Using this, we get

$$N \otimes N \otimes S \cong (N \otimes N)^* \otimes S \cong \mathrm{Hom}(N \otimes N, S)$$

By the universal property of the tensor product, stated in section 2, all maps $N \otimes N \to S$ are associated uniquely to bilinear maps $N \times N \to S$, and then via currying, we get

$$N \otimes N \otimes S \cong \mathrm{Hom}(N \times N, S) \cong \mathrm{Hom}(N, \mathrm{Hom}(N, S))$$

Alternatively, we can bypass the universal property and apply the tensor-hom adjunction (also stated in section 2) directly:

$$N \otimes N \otimes S \cong \mathrm{Hom}(N \otimes N, S) \cong \mathrm{Hom}(N, \mathrm{Hom}(N, S))$$

In either case, we get the following result, which mirrors the case of traditional compositional semantics: The meaning of a transitive verb is a function which takes two nouns (a subject and an object) and produces a sentence, i.e. a vector in S. This interpretation suggests that this model is precisely a generalization of the traditional model. We can make this explicit. Let S be a one-dimensional vector space with basis vector $\{1\}$. We interpret 1 as "true" and 0 (the origin) as "false". In this case, the verb "likes" takes the form

$$\text{likes} = \sum_{i,j} v_i \otimes s_{ij} \otimes w_j$$

where $s_{ij} = 1$ if and only if v_i likes w_j, and 0 otherwise. Now, let the vectors $\{v_i\}$ and $\{w_j\}$ in the above sum range over all nouns in our corpus, i.e. we let all nouns be a basis for the vector space N. We can do this without any mathematical consequences because the corpus is necessarily finite. Also, note

that, by basic properties of the inner product, if x and y are basis elements, then $\langle x, y\rangle = 1$ if $x = y$ and 0 otherwise. We then compute

$$f(\text{Jack} \otimes \text{likes} \otimes \text{Jill}) = \sum_{i,j} \langle \text{Jack}, v_i \rangle s_{ij} \langle w_j, \text{Jill} \rangle$$
$$= s_{\text{Jack,Jill}} = \begin{cases} 1, & \text{Jack likes Jill} \\ 0, & \text{otherwise} \end{cases}$$

Therefore, we achieve the exact same result as we would in the case of traditional compositional semantics, namely that the meaning of "Jack likes Jill" is the truth value of the sentence, and the meaning of "likes" is a function which assigns to two nouns, in a given order, a truth value corresponding to the truth of the resulting sentence.

Since functions are uniquely determined by their input and output, we see that in the case where S is a one-dimensional vector space like above, we recover the traditional model exactly. In this way, the distributional compositional model constructed in this paper is a proper generalization of the traditional one, which allows us the freedom to alter the vector space in which grammatical sentences live (namely by changing the basis) according to what information we would like to extract from the sentence.

9 References

1. Coecke, Bob; Clark, Stephen; Sadrzadeh, Mehrnoosh: Mathematical Foundations for a Compositional Distributional Model of Meaning

2. Coecke, Bob; Grefenstette, Edward; Sadrzadeh, Mehrnoosh: Lambek vs. Lambek: Functorial Vector Space Semantics and String Diagrams for Lambek Calculus