Journal of Artificial Intelligence Research 44 (2012) 533-585 Submitted 03/12; published 07/12

Domain and Function: A Dual-Space Model of Semantic Relations and Compositions

Peter D. Turney PETER.TURNEY@NRC-CNRC.GC.CA
National Research Council Canada
Ottawa, Ontario, Canada, K1A 0R6

Abstract

Given appropriate representations of the semantic relations between *carpenter* and *wood* and between *mason* and *stone* (for example, vectors in a vector space model), a suitable algorithm should be able to recognize that these relations are highly similar (*carpenter* is to *wood* as *mason* is to *stone*; the relations are analogous). Likewise, with representations of *dog*, *house*, and *kennel*, an algorithm should be able to recognize that the semantic composition of *dog* and *house*, *dog house*, is highly similar to *kennel* (*dog house* and *kennel* are synonymous). It seems that these two tasks, recognizing relations and compositions, are closely connected. However, up to now, the best models for relations are significantly different from the best models for compositions. In this paper, we introduce a dual-space model that unifies these two tasks. This model matches the performance of the best previous models for relations and compositions. The dual-space model consists of a space for measuring domain similarity and a space for measuring function similarity. *Carpenter* and *wood* share the same domain, the domain of *carpentry*. *Mason* and *stone* share the same domain, the domain of *masonry*. *Carpenter* and *mason* share the same function, the function of *artisans*. *Wood* and *stone* share the same function, the function of *materials*. In the composition *dog house*, *kennel* has some domain overlap with both *dog* and *house* (the domains of *pets* and *buildings*). The function of *kennel* is similar to the function of *house* (the function of *shelters*). By combining domain and function similarities in various ways, we can model relations, compositions, and other aspects of semantics.

1. Introduction

The *distributional hypothesis* is that words that occur in similar contexts tend to have similar meanings (Harris, 1954; Firth, 1957). Many vector space models (VSMs) of semantics use a word–context matrix to represent the distribution of words over contexts, capturing the intuition behind the distributional hypothesis (Turney & Pantel, 2010). VSMs have achieved impressive results at the level of individual words (Rapp, 2003), but it is not clear how to extend them to the level of phrases, sentences, and beyond. For example, we know how to represent *dog* and *house* with vectors, but how should we represent *dog house*?

One approach to representing *dog house* is to treat it as a unit, the same way we handle individual words. We call this the *holistic* or *noncompositional* approach to representing phrases. The holistic approach may be suitable for some phrases, but it does not scale up. With a vocabulary of N individual words, we can have N^2 two-word phrases, N^3 three-word phrases, and so on. Even with a very large corpus of text, most of these possible phrases will never appear in the corpus. People are continually inventing new phrases, and we are able to understand these new phrases although we have never heard them before; we are able to infer the meaning of a new phrase by *composition* of the meanings of the component words. This scaling problem could be viewed as an issue of data sparsity, but it is better to think of it as a problem of *linguistic creativity* (Chomsky, 1975; Fodor & Lepore, 2002). To master natural language, algorithms must be able to represent phrases by composing representations of individual words. We cannot treat all n-grams ($n > 1$) the way we treat unigrams (individual words). On the other hand, the holistic approach is ideal for idiomatic expressions (e.g., *kick the bucket*) for which the meaning cannot be inferred from the component words.

The creativity and novelty of natural language require us to take a compositional approach to the majority of the n-grams that we encounter. Suppose we have vector representations of *dog* and *house*. How can we compose these representations to represent *dog house*? One strategy is to represent *dog house* by the average of the vectors for *dog* and *house* (Landauer & Dumais, 1997). This simple proposal actually works, to a limited degree (Mitchell & Lapata, 2008, 2010). However *boat house* and *house boat* would be represented by the same average vector, yet they have different meanings. Composition by averaging does not deal with the *order sensitivity* of phrase meaning. Landauer (2002) estimates that 80% of the meaning of English text comes from word choice and the remaining 20% comes from word order.

Similar issues arise with the representation of semantic relations. Given vectors for *carpenter* and *wood*, how can we represent the semantic relations between *carpenter* and *wood*? We can treat *carpenter*:*wood* as a unit and search for paraphrases of the relations between *carpenter* and *wood* (Turney, 2006b). In a large corpus, we could find phrases such as *the carpenter cut the wood*, *the carpenter used the wood*, and *wood for the carpenter*. This variation of the holistic approach can enable us to recognize that the semantic relations between *carpenter* and *wood* are highly similar to the relations between *mason* and *stone*. However, the holistic approach to semantic relations suffers from the same data sparsity and linguistic creativity problems as the holistic approach to semantic composition.

We could represent the relation between *carpenter* and *wood* by averaging their vectors. This might enable us to recognize that *carpenter* is to *wood* as *mason* is to *stone*, but it would incorrectly suggest that *carpenter* is to *wood* as *stone* is to *mason*. The problem of order sensitivity arises with semantic relations just as it arose with semantic composition.

Many ideas have been proposed for composing vectors (Landauer & Dumais, 1997; Kintsch, 2001; Mitchell & Lapata, 2010). Erk and Padó (2008) point out two problems that are common to several of these proposals. First, often they do not have the *adaptive capacity* to represent the variety of possible syntactic relations in a phrase. For example, in the phrase *a horse draws*, *horse* is the subject of the verb *draws*, whereas it is the object of the verb in the phrase *draws a horse*. The composition of the vectors for *horse* and *draws* must be able to adapt to a variety of syntactic contexts in order to properly model the given phrases. Second, a single vector is too weak to handle a long phrase, a sentence, or a document. A single vector "can only encode a fixed amount of structural information if its dimensionality is fixed, but there is no upper limit on sentence length, and hence on the amount of structure to be encoded" (Erk & Padó, 2008, p. 898). A fixed dimensionality does not allow *information scalability*.

Simple (unweighted) averaging of vectors lacks *adaptive capacity*, because it treats all kinds of composition in the same way; it does not have the flexibility to represent different modes of composition. A good model must have the capacity to adapt to different situations.

For example, with weighted averaging, the weights can be tuned for different syntactic contexts (Mitchell & Lapata, 2008, 2010).

Information scalability means that the size of semantic representations should grow in proportion to the amount of information that they are representing. If the size of the representation is fixed, eventually there will be information loss. On the other hand, the size of representations should not grow exponentially.

One case where the problem of information scalability arises is with approaches that map multiple vectors into a single vector. For example, if we represent *dog house* by adding the vectors for *dog* and *house* (mapping two vectors into one), there may be information loss. As we increase the number of vectors that are mapped into a single vector, we will eventually reach a point where the single vector can no longer contain the information from the multiple vectors. This problem can be avoided if we do not try to map multiple vectors into a single vector.

Suppose we have a k-dimensional vector with floating point elements of b bits each. Such a vector can hold at most kb bits of information. Even if we allow b to grow, if k is fixed, we will eventually have information loss. In a vector space model of semantics, the vectors have some resistance to noise. If we perturb a vector with noise below some threshold ϵ, there is no significant change in the meaning that it represents. Therefore we should think of the vector as a hypersphere with a radius of ϵ, rather than a point. We may also put bounds $[-r, +r]$ on the range of the values of the elements in the vector.[1] There is a finite number N of hyperspheres of radius ϵ that can be packed into a bounded k-dimensional space (Conway & Sloane, 1998). According to information theory, if we have a finite set of N messages, then we need at most $\log_2(N)$ bits to encode a message. Likewise, if we have a finite set of N vectors, then a vector represents at most $\log_2(N)$ bits of information. Therefore the information capacity of a single vector in bounded k-dimensional space is limited to $\log_2(N)$ bits.

Past work suggests that recognizing relations and compositions are closely connected tasks (Kintsch, 2000, 2001; Mangalath, Quesada, & Kintsch, 2004). The goal of our research is a unified model that can handle both compositions and relations, while also resolving the issues of linguistic creativity, order sensitivity, adaptive capacity, and information scalability. These considerations have led us to a dual-space model, consisting of a domain space for measuring domain similarity (i.e., topic, subject, or field similarity) and a function space for measuring function similarity (i.e., role, relationship, or usage similarity).

In an analogy $a\!:\!b\!::\!c\!:\!d$ (a is to b as c is to d; for example, *traffic* is to *street* as *water* is to *riverbed*), a and b have relatively high domain similarity (*traffic* and *street* come from the domain of *transportation*) and c and d have relatively high domain similarity (*water* and *riverbed* come from the domain of *hydrology*). On the other hand, a and c have relatively high function similarity (*traffic* and *water* have similar roles in their respective domains; they are both things that *flow*) and b and d have relatively high function similarity (*street* and *riverbed* have similar roles in their respective domains; they are both things that *carry* things that flow). By combining domain and function similarity in appropriate ways, we

1. In models where the vectors are normalized to unit length (e.g., models that use cosine to measure similarity), the elements must lie within the range $[-1, +1]$. If any element is outside this range, then the length of the vector will be greater than one. In general, floating point representations have minimum and maximum values.

can recognize that the semantic relations between *traffic* and *street* are analogous to the relations between *water* and *riverbed*.

For semantic composition, the appropriate way to combine similarities may depend on the syntax of the composition. Let's focus on noun-modifier composition as an example. In the noun-modifier phrase ab (for instance, *brain doctor*), the head noun b (*doctor*) is modified by an adjective or noun a (*brain*). Suppose we have a word c (*neurologist*) that is synonymous with ab. The functional role of the noun-modifier phrase ab is determined by the head noun b (a *brain doctor* is a kind of *doctor*) and b has a relatively high degree of function similarity with c (*doctor* and *neurologist* both function as *doctors*). Both a and b have a high degree of domain similarity with c (*brain*, *doctor*, and *neurologist* all come from the domain of *clinical neurology*). By combining domain and function similarity, we can recognize that *brain doctor* is synonymous with *neurologist*.

Briefly, the proposal is to compose similarity measures instead of composing vectors. That is, we apply various mathematical functions to combine cosine similarity measures, instead of applying the functions directly to the vectors. This addresses the information loss problem, because we preserve the vectors for the individual component words. (We do not map multiple vectors into a single vector.) Since we have two different spaces, we also have flexibility to address the problem of adaptive capacity.[2] This model is compositional, so it resolves the linguistic creativity problem. We deal with order sensitivity by combining similarity measures in ways that recognize the effects of word order.

It might be argued that what we present here is *not* a model of semantic composition, but a way to compare the words that form two phrases in order to derive a measure of similarity of the phrases. For example, in Section 4.3 we derive a measure of similarity for the phrases *environment secretary* and *defence minister*, but we do not actually provide a representation for the phrase *environment secretary*. On the other hand, most past work on the problem of semantic composition (reviewed in Section 2.1) yields a representation for the composite phrase *environment secretary* that is different from the union of the representations of the component words, *environment* and *secretary*.

This argument is based on the assumption that the goal of semantic composition is to create a single, general-purpose, stand-alone representation of a phrase, as a composite, distinct from the union of the representations of the component words. This assumption is not necessary and our approach does not use this assumption. We believe that this assumption has held back progress on the problem of semantic composition.

We argue that what we present here *is* a model of semantic composition, but it is composition of similarities, not composition of vectors. Vectors can represent individual words, but similarities inherently represent relations between two (or more) things. Composing vectors can yield a stand-alone representation of a phrase, but composing similarities necessarily yields a linking structure that connects a phrase to other phrases. Similarity composition does not result in a stand-alone representation of a phrase, but practical applications do not require stand-alone representations. Whatever practical tasks can be performed with stand-alone representations of phrases, we believe can be performed equally well (or better) with similarity composition. We discuss this issue in more depth in Section 6.

2. Two similarity spaces give us more options for similarity composition than one space, just as two types of characters (0 and 1) give us more options for generating strings than one type of character (0 alone).

The next section surveys related work on the modeling of semantic composition and semantic relations. Section 3 describes how we build *domain* and *function* space. To test the hypothesis that there is value in having two separate spaces, we also create *mono* space, which is the merger of the domain and function spaces. We then present four sets of experiments with the dual-space model in Section 4. We evaluate the dual-space approach with multiple-choice analogy questions from the SAT (Turney, 2006b), multiple-choice noun-modifier composition questions derived from WordNet (Fellbaum, 1998), phrase similarity rating problems (Mitchell & Lapata, 2010), and similarity versus association problems (Chiarello, Burgess, Richards, & Pollock, 1990). We discuss the experimental results in Section 5. Section 6 considers some theoretical questions about the dual-space model. Limitations of the model are examined in Section 7. Section 8 concludes.

This paper assumes some familiarity with vector space models of semantics. For an overview of semantic VSMs, see the papers in the *Handbook of Latent Semantic Analysis* (Landauer, McNamara, Dennis, & Kintsch, 2007), the review in Mitchell and Lapata's (2010) paper, or the survey by Turney and Pantel (2010).

2. Related Work

Here we examine related work with semantic composition and relations. In the introduction, we mentioned four problems with semantic models, which yield four desiderata for a semantic model:

1. **Linguistic creativity:** The model should be able to handle phrases (in the case of semantic composition) or word pairs (in the case of semantic relations) that it has never seen before, when it is familiar with the component words.
2. **Order sensitivity:** The model should be sensitive to the order of the words in a phrase (for composition) or a word pair (for relations), when the order affects the meaning.
3. **Adaptive capacity:** For phrases, the model should have the flexibility to represent different kinds of syntactic relations. For word pairs, the model should have the flexibility to handle a variety of tasks, such as measuring the degree of *relational* similarity between two pairs (see Section 4.1) versus measuring the degree of *phrasal* similarity between two pairs (see Section 4.3).
4. **Information scalability:** For phrases, the model should scale up with neither loss of information nor exponential growth in representation size as the number of component words in the phrases increases. For n-ary semantic relations (Turney, 2008a), the model should scale up with neither loss of information nor exponential growth in representation size as n, the number of terms in the relations, increases.

We will review past work in the light of these four considerations.

2.1 Semantic Composition

Let ab be a phrase, such as a noun-modifier phrase, and assume that we have vectors $\mathbf{a}$ and $\mathbf{b}$ that represent the component words a and b. One of the earliest proposals for semantic composition is to represent ab by the vector $\mathbf{c}$ that is the average of $\mathbf{a}$ and $\mathbf{b}$ (Landauer &

Dumais, 1997). If we are using a cosine measure of vector similarity, taking the average of a set of vectors (or their centroid) is the same as adding the vectors, $\mathbf{c} = \mathbf{a}+\mathbf{b}$. Vector addition works relatively well in practice (Mitchell & Lapata, 2008, 2010), although it lacks order sensitivity, adaptive capacity, and information scalability. Regarding order sensitivity and adaptive capacity, Mitchell and Lapata (2008, 2010) suggest using weights, $\mathbf{c} = \alpha\mathbf{a}+\beta\mathbf{b}$, and tuning the weights to different values for different syntactic relations. In their experiments (Mitchell & Lapata, 2010), weighted addition performed better than unweighted addition.

Kintsch (2001) proposes a variation of additive composition in which $\mathbf{c}$ is the sum of $\mathbf{a}$, $\mathbf{b}$, and selected neighbours $\mathbf{n}_i$ of $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{c} = \mathbf{a}+\mathbf{b}+\sum_i \mathbf{n}_i$. The neighbours are vectors for other words in the given vocabulary (i.e., other rows in the given word–context matrix). The neighbours are chosen in a manner that attempts to address order sensitivity and adaptive capacity, but there is still a problem with information scalability due to fixed dimensionality. Utsumi (2009) presents a similar model, but with a different way of selecting neighbours. Mitchell and Lapata (2010) found that a simple additive model peformed better than an additive model that included neighbours.

Mitchell and Lapata (2008, 2010) suggest element-wise multiplication as a composition operation, $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$, where $c_i = a_i \cdot b_i$. Like vector addition, element-wise multiplication suffers from a lack of order sensitivity, adaptive capacity, and information scalability. Nonetheless, in an experimental evaluation of seven compositional models and two non-compositional models, element-wise multiplication had the best performance (Mitchell & Lapata, 2010).

Another approach is to use a tensor product for composition (Smolensky, 1990; Aerts & Czachor, 2004; Clark & Pulman, 2007; Widdows, 2008), such as the outer product, $\mathbf{C} = \mathbf{a} \otimes \mathbf{b}$. The outer product of two vectors ($\mathbf{a}$ and $\mathbf{b}$), each with n elements, is an $n \times n$ matrix ($\mathbf{C}$). The outer product of three vectors is an $n \times n \times n$ third-order tensor. This results in an information scalability problem: The representations grow exponentially large as the phrases grow longer.[3] Furthermore, the outer product did not perform as well as element-wise multiplication in Mitchell and Lapata's (2010) experiments. Recent work with tensor products (Clark, Coecke, & Sadrzadeh, 2008; Grefenstette & Sadrzadeh, 2011) has attempted to address the issue of information scalability.

Circular convolution is similar to the outer product, but the outer product matrix is compressed back down to a vector, $\mathbf{c} = \mathbf{a} \circledast \mathbf{b}$ (Plate, 1995; Jones & Mewhort, 2007). This avoids information explosion, but it results in information loss. Circular convolution performed poorly in Mitchell and Lapata's (2010) experiments.

Baroni and Zamparelli (2010) and Guevara (2010) suggest another model of composition for adjective-noun phrases. The core strategy that they share is to use a few holistic vectors to train a compositional model. With partial least squares regression (PLSR), we can learn a linear model that maps the vectors for the component nouns and adjectives to linear approximations of the holistic vectors for the phrases. The linguistic creativity problem is avoided because the linear model only needs a few holistic vectors for training; there is no need to have holistic vectors for all plausible adjective-noun phrases. Given a phrase that is not in the training data, the linear model predicts the holistic vector for the phrase, given

3. There are ways to avoid the exponential growth; for example, a third-order tensor with a rank of 1 on all three modes may be compactly encoded by its three component vectors. Kolda and Bader (2009) discuss compact tensor representations.

the component vectors for the adjective and the noun. This works well for adjective-noun phrases, but it is not clear how to generalize it to other parts of speech or to longer phrases.

One application for semantic composition is measuring the similarity of phrases (Erk & Padó, 2008; Mitchell & Lapata, 2010). Kernel methods have been applied to the closely related task of identifying paraphrases (Moschitti & Quarteroni, 2008), but the emphasis with kernel methods is on syntactic similarity, rather than semantic similarity.

Neural network models have been combined with vector space models for the task of language modeling (Bengio, Ducharme, Vincent, & Jauvin, 2003; Socher, Manning, & Ng, 2010; Socher, Huang, Pennington, Ng, & Manning, 2011), with impressive results. The goal of a language model is to estimate the probability of a phrase or to decide which of several phrases is the most likely. VSMs can improve the probability estimates of a language model by measuring the similarity of the words in the phrases and smoothing probabilities over groups of similar words. However, in a language model, words are considered similar to the degree that they can be exchanged without altering the *probability* of a given phrase, without regard to whether the exchange alters the *meaning* of the phrase. This is like function similarity, which measures the degree to which words have similar functional roles, but these language models are missing anything like domain similarity.

Erk and Padó (2008) present a model that is similar to ours in that it has two parts, a vector space for measuring similarity and a model of selectional preferences. Their vector space is similar to domain space and their model of selectional preferences plays a role similar to function space. An individual word a is represented by a triple, $A = \langle \mathbf{a}, R, R^{-1} \rangle$, consisting of the word's vector, $\mathbf{a}$, its selectional preferences, R, and its inverse selectional preferences, R^{-1}. A phrase ab is represented by a pair of triples, $\langle A', B' \rangle$. The triple A' is a modified form of the triple A that represents the individual word a. The modifications adjust the representation to model how the meaning of a is altered by its relation to b in the phrase ab. Likewise, the triple B' is a modified form of the triple B that represents b, such that B' takes into account how a affects b.

When A is transformed to A' to represent the influence of b on the meaning of a, the vector $\mathbf{a}$ in A is transformed to a new vector $\mathbf{a}'$ in A'. Let $\mathbf{r}_b$ be a vector that represents the typical words that are consistent with the selectional preferences of b. The vector $\mathbf{a}'$ is the composition of $\mathbf{a}$ with $\mathbf{r}_b$. Erk and Padó (2008) use element-wise multiplication for composition, $\mathbf{a}' = \mathbf{a} \odot \mathbf{r}_b$. The intention is to make $\mathbf{a}$ more like a typical vector $\mathbf{x}$ that would be expected for a phrase xb. Likewise, for $\mathbf{b}'$ in B', we have $\mathbf{b}' = \mathbf{b} \odot \mathbf{r}_a$

Erk and Padó's (2008) model and related models (Thater, Fürstenau, & Pinkal, 2010) address linguistic creativity, order sensitivity, adaptive capacity, and information scalability, but they are not suitable for measuring the similarity of semantic relations. Consider the analogy *traffic* is to *street* as *water* is to *riverbed*. Let $\langle A', B' \rangle$ represent *traffic*:*street* and let $\langle C', D' \rangle$ represent *water*:*riverbed*. The transformation of A, B, C, and D to A', B', C', and D' reinforces the connection between *traffic* and *street* and between *water* and *riverbed*, but it does not help us recognize the relational similarity between *traffic*:*street* and *water*:*riverbed*. Of course, these models were not designed for relational similarity, so this is not surprising. However, the goal here is to find a unified model that can handle both compositions and relations.

2.2 Semantic Relations

For semantic relations, we can make some general observations about order sensitivity. Let $a:b$ and $c:d$ be two word pairs and let $\mathrm{sim_r}(a:b, c:d) \in \Re$ be a measure of the degree of similarity between the relations of $a:b$ and $c:d$. If $a:b::c:d$ is a good analogy, then $\mathrm{sim_r}(a:b, c:d)$ will have a relatively high value. In general, a good model of relational similarity should respect the following equalities and inequalities:

$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) = \mathrm{sim_r}(b\!:\!a, d\!:\!c) \tag{1}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) = \mathrm{sim_r}(c\!:\!d, a\!:\!b) \tag{2}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) \neq \mathrm{sim_r}(a\!:\!b, d\!:\!c) \tag{3}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) \neq \mathrm{sim_r}(a\!:\!d, c\!:\!b) \tag{4}$$

For example, given that *carpenter*:*wood* and *mason*:*stone* make a good analogy, it follows from Equation 1 that *wood*:*carpenter* and *stone*:*mason* make an equally good analogy. Also, according to Equation 2, *mason*:*stone* and *carpenter*:*wood* make a good analogy. On the other hand, as suggested by Equation 3, *carpenter*:*wood* is *not* analogous to *stone*:*mason*. Likewise, as indicated by Equation 4, it is a *poor* analogy to assert that *carpenter* is to *stone* as *mason* is to *wood*.

Rosario and Hearst (2001) present an algorithm for classifying word pairs according to their semantic relations. They use a lexical hierarchy to map word pairs to feature vectors. Any classification scheme implicitly tell us something about similarity. Two word pairs that are in the same semantic relation class are implicitly more relationally similar than two word pairs in different classes. When we consider the relational similarity that is implied by Rosario and Hearst's (2001) algorithm, we see that there is a problem of order sensitivity: Equation 4 is violated.

Let $\mathrm{sim_h}(x, y) \in \Re$ be a measure of the degree of hierarchical similarity between the words x and y. If $\mathrm{sim_h}(x, y)$ is relatively high, then x and y share a common hypernym relatively close to them in the given lexical hierarchy. In essence, the intuition behind Rosario and Hearst's (2001) algorithm is, if both $\mathrm{sim_h}(a, c)$ and $\mathrm{sim_h}(b, d)$ are high, then $\mathrm{sim_r}(a:b, c:d)$ should also be high. That is, if $\mathrm{sim_h}(a, c)$ and $\mathrm{sim_h}(b, d)$ are high enough, then $a\!:\!b$ and $c\!:\!d$ should be assigned to the same relation class.

For example, consider the analogy *mason* is to *stone* as *carpenter* is to *wood*. The common hypernym of *mason* and *carpenter* is *artisan*; we can see that $\mathrm{sim_h}(\mathit{mason}, \mathit{carpenter})$ is high. The common hypernym of *stone* and *wood* is *material*; hence $\mathrm{sim_h}(\mathit{stone}, \mathit{wood})$ is high. It seems that a good analogy is indeed characterized by high values for $\mathrm{sim_h}(a, c)$ and $\mathrm{sim_h}(b, d)$. However, the symmetry of $\mathrm{sim_h}(x, y)$ leads to a problem. If $\mathrm{sim_h}(b, d)$ is high, then $\mathrm{sim_h}(d, b)$ must also be high, but this implies that $\mathrm{sim_r}(a\!:\!d, c\!:\!b)$ is high. That is, we incorrectly conclude that *mason* is to *wood* as *carpenter* is to *stone* (see Equation 4).

Some later work with classifying semantic relations has used different algorithms, but the same underlying intuition about hierarchical similarity (Rosario, Hearst, & Fillmore, 2002; Nastase & Szpakowicz, 2003; Nastase, Sayyad-Shirabad, Sokolova, & Szpakowicz, 2006). We use a similar intuition here, since similarity in function space is closely related to hierarchical similarity, $\mathrm{sim_h}(x, y)$, as we will see later (Section 4.4). However, including domain space in the relational similarity measure saves us from violating Equation 4.

Let $\mathrm{sim_f}(x, y) \in \Re$ be function similarity as measured by the cosine of vectors $\mathbf{x}$ and $\mathbf{y}$ in function space. Let $\mathrm{sim_d}(x, y) \in \Re$ be domain similarity as measured by the cosine of vectors $\mathbf{x}$ and $\mathbf{y}$ in domain space. Like past researchers (Rosario & Hearst, 2001; Rosario et al., 2002; Nastase & Szpakowicz, 2003; Veale, 2004; Nastase et al., 2006), we look for high values of $\mathrm{sim_f}(a, c)$ and $\mathrm{sim_f}(b, d)$ as indicators that $\mathrm{sim_r}(a\!:\!b, c\!:\!d)$ should be high, but we also look for high values of $\mathrm{sim_d}(a, b)$ and $\mathrm{sim_d}(c, d)$. Continuing the previous example, we do *not* conclude that *mason* is to *wood* as *carpenter* is to *stone*, because *wood* does not belong in the domain of *masonry* and *stone* does not belong in the domain of *carpentry*.

Let D be a determiner (e.g., *the*, *a*, *an*). Hearst (1992) showed how patterns of the form "D X such as D Y" ("a bird such as a crow") or "D Y is a kind of X" ("the crow is a kind of bird") can be used to infer that X is a hypernym of Y (*bird* is a hypernym of *crow*). A pair–pattern matrix is a VSM in which the rows are word pairs and the columns are various "X ... Y" patterns. Turney, Littman, Bigham, and Shnayder (2003) demonstrated that a pair–pattern VSM can be used to measure relational similarity. Suppose we have a pair-pattern matrix $\mathbf{X}$ in which the word pair $a\!:\!b$ corresponds to the row vector $\mathbf{x}_i$ and $c\!:\!d$ corresponds to $\mathbf{x}_j$. The approach is to measure the relational similarity $\mathrm{sim_r}(a\!:\!b, c\!:\!d)$ by the cosine of $\mathbf{x}_i$ and $\mathbf{x}_j$.

At first the patterns in these pair–pattern matrices were generated by hand (Turney et al., 2003; Turney & Littman, 2005), but later work (Turney, 2006b) used automatically generated patterns. Other authors have used variations of this technique (Nakov & Hearst, 2006, 2007; Davidov & Rappoport, 2008; Bollegala, Matsuo, & Ishizuka, 2009; Ó Séaghdha & Copestake, 2009). All of these models suffer from the linguistic creativity problem. Because the models are noncompositional (holistic), they cannot scale up to handle the huge number of possible pairs. Even the largest corpus cannot contain all the pairs that a human speaker might use in daily conversation.

Turney (2006b) attempted to handle the linguistic creativity problem within a holistic model by using synonyms. For example, if a corpus does not contain *traffic* and *street* within a certain window of text, perhaps it might contain *traffic* and *road*. If it does not contain *water* and *riverbed*, perhaps it has *water* and *channel*. However, this is at best a partial solution. Turney's (2006b) algorithm required nine days to process 374 multiple choice SAT analogy questions. Using the dual-space model, without specifying in advance what word pairs it might face, we can answer the 374 questions in a few seconds (see Section 4.1). Compositional models scale up better than holistic models.

Mangalath et al. (2004) presented a model for semantic relations that represents word pairs with vectors of ten abstract relational categories, such as *hyponymy*, *meronymy*, *taxonomy*, and *degree*. The approach is to construct a kind of second-order vector space in which the elements of the vectors are degrees of similarity, calculated from cosines with a first-order word–context matrix.

For instance, *carpenter*:*wood* can be represented by a second-order vector composed of ten cosines calculated from first-order vectors. In this second-order vector, the value of the element corresponding to, say, *meronymy* would be the cosine of two first-order vectors, $\mathbf{x}$ and $\mathbf{y}$. The vector $\mathbf{x}$ would be the sum of the first-order vectors for *carpenter* and *wood*. The vector $\mathbf{y}$ would be the sum of several vectors for words that are related to *meronymy*, such as *part*, *whole*, *component*, *portion*, *contains*, *constituent*, and *segment*. The cosine of $\mathbf{x}$ and $\mathbf{y}$ would indicate the degree to which *carpenter* and *wood* are related to *meronymy*.

Mangalath et al.'s (2004) model suffers from information scalability and order sensitivity problems. Information loss takes place when the first-order vectors are summed and also when the high-dimensional first-order space is reduced to a ten-dimensional second-order space. The order sensitivity problem is that the second-order vectors violate Equation 3, because the pairs $c\!:\!d$ and $d\!:\!c$ are represented by the same second-order vector.

A natural proposal is to represent a word pair $a : b$ the same way we would represent a phrase ab. That is, whatever compositional model we have for phrases could also be applied to word pairs. However any problems that the compositional model has with order sensitivity or information scalability carry over to word pairs. For example, if we represent $a : b$ by $\mathbf{c} = \mathbf{a} + \mathbf{b}$ or $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$, then we violate Equation 3, because $\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}$ and $\mathbf{a} \odot \mathbf{b} = \mathbf{b} \odot \mathbf{a}$.

3. Three Vector Spaces

In this section, we describe three vector space models. All three spaces consist of word–context matrices, in which the rows correspond to words and the columns correspond to the contexts in which the words occur. The differences among the three spaces are in the kinds of contexts. Domain space uses nouns for context, function space uses verb-based patterns for context, and mono space is a merger of the domain and function contexts. Mono space was created in order to test the hypothesis that it is useful to separate the domain and function spaces; mono space serves as a baseline.

3.1 Constructing the Word–Context Matrices

Building the three spaces involves a series of steps. There are three main steps, each of which has a few substeps. The first and last steps are the same for all three spaces; the differences in the spaces are the result of differences in the second step.

1. **Find terms in contexts:** input: a corpus and a lexicon, output: terms in contexts.
    - 1.1. Extract terms from the lexicon and find their frequencies in the corpus.
    - 1.2. Select all terms above a given frequency as candidate rows for the frequency matrix.
    - 1.3. For each selected term, find phrases in the corpus that contain the term within a given window size.
    - 1.4. Use a tokenizer to split the phrases into tokens.
    - 1.5. Use a part-of-speech tagger to tag the tokens in the phrases.
2. **Build a term–context frequency matrix:** input: terms in contexts, output: a sparse frequency matrix.
    - 2.1. Convert the tagged phrases into contextual patterns (candidate columns).
    - 2.2. For each contextual pattern, count the number of terms (candidate rows) that generated the pattern and rank the patterns in descending order of their counts.
    - 2.3. Select the top n_c contextual patterns as the columns of the matrix.

2.4. From the initial set of rows (from Step 1.2), drop any row that does not match any of the top n_c contextual patterns, yielding the final set of n_r rows.

2.5. For each row (term) and each column (contextual pattern), count the number of phrases (from Step 1.5) containing the given term and matching the given pattern, and output the resulting numbers as a sparse frequency matrix.

3. **Weight the elements and smooth the matrix:** input: a sparse frequency matrix, output: the singular value decomposition (SVD) of the weighted matrix.

   3.1. Convert the raw frequencies to positive pointwise mutual information (PPMI) values.

   3.2. Apply SVD to the PPMI matrix and output the SVD component matrices.

The input corpus in Step 1 is a collection of web pages gathered from university websites by a webcrawler.[4] The corpus contains approximately 5×10^{10} words, which comes to about 280 gigabytes of plain text. To facilitate finding term frequencies and sample phrases, we indexed this corpus with the Wumpus search engine (Büttcher & Clarke, 2005).[5] The rows for the matrices were selected from terms (words and phrases) in the WordNet lexicon.[6] We found that selecting terms from WordNet resulted in subjectively higher quality than simply selecting terms with high corpus frequencies.

In Step 1.1, we extract all unique words and phrases (n-grams) from the *index.sense* file in WordNet 3.0, skipping n-grams that contain numbers (only letters, hyphens, and spaces are allowed in the n-grams). We find the n-gram corpus frequencies by querying Wumpus with each n-gram. All n-grams with a frequency of at least 100 and at least 2 characters are candidate rows in Step 1.2. For each selected n-gram, we query Wumpus to find a maximum of 10,000 phrases in Step 1.3.[7] The phrases are limited to a window of 7 words to the left of the n-gram and 7 words to the right, for a total window size of $14 + n$ words. We use OpenNLP 1.3.0 to tokenize and part-of-speech tag the phrases (Steps 1.4 and 1.5).[8] The tagged phrases come to about 46 gigabytes.[9]

In Step 2.1, we generate contextual patterns from the part-of-speech tagged phrases. Different kinds of patterns are created for the three different kinds of spaces. The details of this step are given in the following subsections. Each phrase may yield several patterns. The three spaces each have more than 100,000 rows, with a maximum of 10,000 phrases per row and several patterns per phrase. This can result in millions of distinct patterns, so we filter the patterns in Steps 2.2 and 2.3. We select the top n_c patterns that are shared by the largest number of rows. Given the large number of patterns, they may not all fit in RAM. To work with limited RAM, we use the Linux *sort* command, which is designed to efficiently sort files that are too large to fit in RAM. For each row, we make a file of the distinct patterns generated by that row. We then concatenate all of the files for all of

4. The corpus was collected by Charles Clarke at the University of Waterloo.
5. Wumpus is available at http://www.wumpus-search.org/.
6. WordNet is available at http://wordnet.princeton.edu/.
7. The limit of 10,000 phrases per n-gram is required to make Wumpus run in a tolerable amount of time. Finding phrases is the most time-consuming step in the construction of the spaces. We use a solid-state drive (SSD) to speed up this step.
8. OpenNLP is available at http://incubator.apache.org/opennlp/.
9. The tagged phrases are available from the author on request.

the rows and alphabetically sort the patterns in the concatenated file. In the sorted file, identical patterns are adjacent, which makes it easy to count the number of occurrences of each pattern. After counting, a second sort operation yields a ranked list of patterns, from which we select the top n_c.

It is possible that some of the candidate rows from Step 1.2 might not match any of the patterns from Step 2.3. These rows would be all zeros in the matrix, so we remove them in Step 2.4. Finally, we output a sparse frequency matrix $\mathbf{F}$ with n_r rows and n_c columns. If the i-th row corresponds to the n-gram w_i and the j-th column corresponds to the contextual pattern c_j, then the value of the element f_{ij} in $\mathbf{F}$ is the number of phrases containing w_i (from Step 1.5) that generate the pattern c_j (in Step 2.1). In Step 3.2, we use SVDLIBC 1.34 to calculate the singular value decomposition, so the format of the output sparse matrix in Step 2.5 is chosen to meet the requirements of SVDLIBC.[10]

In Step 3.1, we apply positive pointwise mutual information (PPMI) to the sparse frequency matrix $\mathbf{F}$. This is a variation of pointwise mutual information (PMI) (Church & Hanks, 1989; Turney, 2001) in which all PMI values that are less than zero are replaced with zero (Niwa & Nitta, 1994; Bullinaria & Levy, 2007). Let $\mathbf{X}$ be the matrix that results when PPMI is applied to $\mathbf{F}$. The new matrix $\mathbf{X}$ has the same number of rows and columns as the raw frequency matrix $\mathbf{F}$. The value of an element x_{ij} in $\mathbf{X}$ is defined as follows:

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{5}$$

$$p_{i*} = \frac{\sum_{j=1}^{n_c} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{6}$$

$$p_{*j} = \frac{\sum_{i=1}^{n_r} f_{ij}}{\sum_{i=1}^{n_r} \sum_{j=1}^{n_c} f_{ij}} \tag{7}$$

$$\mathrm{pmi}_{ij} = \log\left(\frac{p_{ij}}{p_{i*}p_{*j}}\right) \tag{8}$$

$$x_{ij} = \begin{cases} \mathrm{pmi}_{ij} & \text{if } \mathrm{pmi}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In this definition, p_{ij} is the estimated probability that the word w_i occurs in the context c_j, p_{i*} is the estimated probability of the word w_i, and p_{*j} is the estimated probability of the context c_j. If w_i and c_j are statistically independent, then $p_{ij} = p_{i*}p_{*j}$ (by the definition of independence), and thus pmi_{ij} is zero (since $\log(1) = 0$). The product $p_{i*}p_{*j}$ is what we would expect for p_{ij} if w_i occurs in c_j by pure random chance. On the other hand, if there is an interesting semantic relation between w_i and c_j, then we should expect p_{ij} to be larger than it would be if w_i and c_j were indepedent; hence we should find that $p_{ij} > p_{i*}p_{*j}$, and thus pmi_{ij} is positive. If the word w_i is unrelated to (or incompatible with) the context c_j, we may find that pmi_{ij} is negative. PPMI is designed to give a high value to x_{ij} when there is an interesting semantic relation between w_i and c_j; otherwise, x_{ij} should have a value of zero, indicating that the occurrence of w_i in c_j is uninformative.

10. SVDLIBC is available at http://tedlab.mit.edu/~dr/svdlibc/.

Finally, in Step 3.2, we apply SVDLIBC to $\mathbf{X}$. SVD decomposes $\mathbf{X}$ into the product of three matrices $\mathbf{U\Sigma V}^\mathsf{T}$, where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form (i.e., the columns are orthogonal and have unit length, $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}$) and $\mathbf{\Sigma}$ is a diagonal matrix of singular values (Golub & Van Loan, 1996). If $\mathbf{X}$ is of rank r, then $\mathbf{\Sigma}$ is also of rank r. Let $\mathbf{\Sigma}_k$, where $k < r$, be the diagonal matrix formed from the top k singular values, and let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the matrices produced by selecting the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ is the matrix of rank k that best approximates the original matrix $\mathbf{X}$, in the sense that it minimizes the approximation errors. That is, $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$ minimizes $\|\hat{\mathbf{X}} - \mathbf{X}\|_F$ over all matrices $\hat{\mathbf{X}}$ of rank k, where $\| \ldots \|_F$ denotes the Frobenius norm (Golub & Van Loan, 1996). The final output is the three matrices, $\mathbf{U}_k$, $\mathbf{\Sigma}_k$, and $\mathbf{V}_k$, that form the truncated SVD, $\hat{\mathbf{X}} = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T}$.

3.2 Domain Space

The intuition behind domain space is that the domain or topic of a word is characterized by the nouns that occur near it. We use a relatively wide window and we ignore the syntactic context in which the nouns appear.

For domain space, in Step 2.1, each tagged phrase generates at most two contextual patterns. The contextual patterns are simply the first noun to the left of the given n-gram (if there is one) and the first noun to the right (if there is one). Since the window size is 7 words on each side of the n-gram, there are usually nouns on both sides of the n-gram. The nouns may be either common nouns or proper nouns. OpenNLP uses the Penn Treebank tags (Santorini, 1990), which include several different categories of noun tags. All of the noun tags begin with a capital N, so we simply extract the first words to the left and right of the n-gram that have tags that begin with N. The extracted nouns are converted to lower case. If the same noun appears on both sides of the n-gram, only one contextual pattern is generated. The extracted patterns are always unigrams; in a noun compound, only the component noun closest to the n-gram is extracted.

Table 1 shows some examples for the n-gram *boat*. Note that the window of 7 words does not count punctuation, so the number of tokens in the window may be greater than the number of words in the window. We can see from Table 1 that the row vector for the n-gram *boat* in the frequency matrix $\mathbf{F}$ will have nonzero values (for example) in the columns for *lake* and *summer* (assuming that these contextual patterns make it through the filtering in Step 2.3).

For Step 2.3, we set n_c to 50,000. In Step 2.4, after we drop rows that are all zero, we are left with n_r equal to 114,297. After PPMI (which sets negative elements to zero) we have 149,673,340 nonzero values, for a matrix density of 2.62%. Table 2 shows the contextual patterns for the first five columns and the last five columns (the columns are in order of their ranks in Step 2.2). The *Count* column of the table gives the number of rows (n-grams) that generate the pattern (that is, these are the counts mentioned in Step 2.2). The last patterns all begin with *c* because they have the same counts and ties are broken by alphabetical order.

	Tagged phrases	Patterns
1	"would/MD visit/VB Big/NNP Lake/NNP and/CC take/VB our/PRP$ **boat/NN** on/IN this/DT huge/JJ beautiful/JJ lake/NN ./. There/EX was/VBD"	"lake"
2	"the/DT large/JJ paved/JJ parking/NN lot/NN in/IN the/DT **boat/NN** ramp/NN area/NN and/CC walk/VB south/RB along/IN the/DT"	"lot" "ramp"
3	"building/VBG permit/NN ./. '/" Anyway/RB ,/, we/PRP should/MD have/VB a/DT **boat/NN** next/JJ summer/NN with/IN skiing/NN and/CC tubing/NN paraphernalia/NNS ./."	"permit" "summer"

Table 1: Examples of Step 2.1 in domain space for the n-gram *boat*. The three tagged phrases generate five contextual patterns.

Column	Pattern	Count
1	"time"	91,483
2	"part"	84,445
3	"years"	84,417
4	"way"	84,172
5	"name"	81,960

Column	Pattern	Count
49,996	"clu"	443
49,997	"co-conspirator"	443
49.998	"conciseness"	443
49,999	"condyle"	443
50,000	"conocer"	443

Table 2: Contextual patterns for the first and last columns in domain space. *CLU* is an abbreviation for Chartered Life Underwriter and other terms, *condyle* is a round bump on a bone where it forms a joint with another bone, and *conocer* is the Spanish verb *to know*, in the sense of being acquainted with a person.

3.3 Function Space

The concept of function space is that the function or role of a word is characterized by the syntactic context that relates it to the verbs that occur near it. We use a more narrow window for function space than domain space, based on the intuition that proximity to a verb is important for determining the functional role of the given word. A distant verb is less likely to characterize the function of the word. We generate relatively complex patterns for function space, to try to capture the syntactic patterns that connect the given word to the nearby verbs.

In Step 2.1, each tagged phrase generates up to six contextual patterns. For a given tagged phrase, the first step is to cut the window down to 3 tokens before the given n-gram and 3 tokens after it. If any of the remaining tokens to the left of the n-gram are punctuation, the punctuation and everything to the left of the punctuation is removed. If any of the remaining tokens to the right of the n-gram are punctuation, the punctuation and everything to the right of the punctuation is removed. Let's call the remaining tagged phrase a truncated tagged phrase.

Next we replace the given n-gram in the truncated tagged phrase with a generic marker,

X. We then simplify the part-of-speech tags by reducing them all to their first character (Santorini, 1990). For example, all of the various verb tags (VB, VBD, VBG, VBN, VBP, VBZ) are reduced to V. If the truncated tagged phrase contains no V tag, it generates zero contextual patterns. If the phrase contains a V tag, then we generate two types of contextual patterns, general patterns and specific patterns.

For the general patterns, the verbs (every token with a V tag) have their tags removed (naked verbs) and all other tokens are reduced to naked tags (tags without words). For the specific patterns, verbs, modals (tokens with M tags), prepositions (tokens with I tags), and *to* (tokens with T tags) have their tags removed and all other tokens are reduced to naked tags. (See Table 3 for examples.)

For both general and specific patterns, to the left of X, we trim any leading naked tags. To the right of X, we trim any trailing naked tags. A T tag can only be *to*, so we replace any remaining naked T tags with *to*. A sequence of N tags (N N or N N N) is likely a compound noun, so we reduce the sequence to a single N.

For a given truncated tagged phrase, we now have two patterns, one general pattern and one specific pattern. If either of these patterns has tokens on both the left and right sides of X, we make two more patterns by duplicating the X and then splitting the pattern at the point between the two Xs. If one of the new patterns does not have a verb, we drop it. Thus we may now have up to three specific patterns and three general patterns for the given truncated tagged phrase. If the specific and general patterns are the same, only one of them is generated.

Table 3 shows some examples for the n-gram *boat*. Note that every pattern must contain the generic marker, X, and at least one verb.

	Truncated tagged phrases	Patterns	Types
1	"the/DT canals/NNS by/IN **boat/NN** and/CC wandering/VBG the/DT"	"X C wandering"	general
		"by X C wandering"	specific
2	"a/DT charter/NN fishing/VBG **boat/NN** captain/NN named/VBN Jim/NNP"	"fishing X N named"	general
		"fishing X"	general
		"X N named"	general
3	"used/VBN from/IN a/DT **boat/NN** and/CC lowered/VBD to/TO"	"used I D X C lowered"	general
		"used I D X"	general
		"X C lowered"	general
		"used from D X C lowered to"	specific
		"used from D X"	specific
		"X C lowered to"	specific

Table 3: Examples of Step 2.1 in function space for the n-gram *boat*. The three truncated tagged phrases generate eleven contextual patterns.

For Step 2.3, we set n_c to 50,000. In Step 2.4, after rows that are all zero are dropped, n_r is 114,101. After PPMI, there are 68,876,310 nonzero values, yielding a matrix density of 1.21%. Table 4 shows the contextual patterns for the first and the last five columns. The last patterns all begin with *s* because they have the same counts and ties are broken by alphabetical order.

Column	Pattern	Count
1	"X is"	94,312
2	"X N is"	82,171
3	"is D X"	79,131
4	"is X"	72,637
5	"X was"	72,497

Column	Pattern	Count
49,996	"since D X N was"	381
49,997	"sinking I D X"	381
49,998	"supplied with X"	381
49,999	"supports D X N of"	381
50,000	"suppressed I D X"	381

Table 4: Contextual patterns for the first and last columns in function space.

The contextual patterns for function space are more complex than the patterns for domain space. The motivation for this greater complexity is the observation that mere proximity is not enough to determine functional roles, although it seems sufficient for determining domains. For example, consider the verb *gives*. If there is a word X that occurs near *gives*, X could be the subject, direct object, or indirect object of the verb. To determine the functional role of X, we need to know which case applies. The syntactic context that connects X to *gives* provides this information. The contextual pattern "X gives" implies that X is the subject, "gives X" implies X is an object, likely the direct object, and "gives to X" suggests that X is the indirect object. Modals and prepositions supply further information about the functional role of X in the context of a given verb. The verb *gives* appears in 43 different contextual patterns (i.e., 43 of the 50,000 columns in function space correspond to syntactic patterns that contain *gives*).

Many of the row vectors in the function space matrix correspond to verbs. It might seem surprising that we can characterize the function of a verb by its syntactic relation to other verbs, but consider an example, such as the verb *run*. The row vector for *run* in the PPMI matrix for function space has 1,296 nonzero values; that is, *run* is characterized by 1,296 different contextual patterns.

Note that appearing in a contextual pattern is different from having a nonzero value for a contextual pattern. The character string for the word *run* appears in 62 different contextual patterns, such as "run out of X". The row vector for the word *run* has nonzero values for 1,296 contextual patterns (columns), such as "had to X".

3.4 Mono Space

Mono space is simply the merger of domain space and function space. For Step 2.3, we take the union of the 50,000 domain space columns and the 50,000 function space columns, resulting in a total n_c of 100,000 columns. In Step 2.4, we have a total n_r of 114,297 rows. The mono matrix after PPMI has 218,222,254 nonzero values, yielding a density of 1.91%. The values in the mono frequency matrix $\mathbf{F}$ equal the corresponding values in the domain and function matrices. Some of the rows in the mono space matrix do not have corresponding rows in the function space matrix. For these rows, the corresponding values are zeros (but there are nonzero elements in these rows, which correspond to values in the domain matrix).

3.5 Summary of the Spaces

Table 5 summarizes the three matrices. In the following four sets of experiments, we use the same three matrices (the domain, function, and mono matrices) in all cases; we do not generate different matrices for each set of experiments. Three of the four sets of experiments involve datasets that have been used in past by other researchers. We made no special effort to ensure that the words in these three datasets have corresponding rows in the three matrices. The intention is that these three matrices should be adequate to handle most applications without any special customization.

Space	Rows (n_r)	Columns (n_c)	Nonzeros (after PPMI)	Density (after PPMI)
domain	114,297	50,000	149,673,340	2.62%
function	114,101	50,000	68,876,310	1.21%
mono	114,297	100,000	218,222,254	1.91%

Table 5: Summary of the three spaces.

3.6 Using the Spaces to Measure Similarity

In the following experiments, we measure the similarity of two terms, a and b, by the cosine of the angle θ between their corresponding row vectors, $\mathbf{a}$ and $\mathbf{b}$:

$$\text{sim}(a, b) = \cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}}{\|\mathbf{a}\|} \cdot \frac{\mathbf{b}}{\|\mathbf{b}\|} \tag{10}$$

The cosine of the angle between two vectors is the inner product of the vectors, after they have been normalized to unit length. The cosine ranges from -1 when the vectors point in opposite directions (θ is 180 degrees) to $+1$ when they point in the same direction (θ is 0 degrees). When the vectors are orthogonal (θ is 90 degrees), the cosine is zero. With raw frequency vectors, which necessarily cannot have negative elements, the cosine cannot be negative, but weighting and smoothing often introduce negative elements. PPMI weighting does not yield negative elements, but truncated SVD can generate negative elements, even when the input matrix has no negative values.

The semantic similarity of two terms is given by the cosine of the two corresponding rows in $\mathbf{U}_k\mathbf{\Sigma}_k^p$ (see Section 3.1). There are two parameters in $\mathbf{U}_k\mathbf{\Sigma}_k^p$ that need to be set. The parameter k controls the number of latent factors and the parameter p adjusts the weights of the factors, by raising the corresponding singular values in $\mathbf{\Sigma}_k^p$ to the power p. The parameter k is well-known in the literature (Landauer et al., 2007), but p is less familiar. The use of p was suggested by Caron (2001). In the following experiments (Section 4), we explore a range of values for p and k.

Suppose we take a word w and list all of the other words in descending order of their cosines with w, using $\mathbf{U}_k\mathbf{\Sigma}_k^p$ to calculate the cosines. When p is high, as we go down the list, the cosines of the nearest neighbours of w decrease slowly. When p is low, they decrease quickly. That is, a high p results in a broad, fuzzy neighbourhood and a low p yields a sharp, crisp neighbourhood. The parameter p controls the *sharpness* of the similarity measure.

To reduce the running time of SVDLIBC, we limit the number of singular values to 1500, which usually results in less than 1500 singular values. For example, the SVD for domain space has 1477 singular values. As long as k is not greater than 1477, we can experiment with a range of k values without rerunning SVDLIBC. We can generate $\mathbf{U}_k \mathbf{\Sigma}_k^p$ from $\mathbf{U}_{1477} \mathbf{\Sigma}_{1477}^p$ by simply deleting the $1477 - k$ columns with the smallest singular values.

In the experiments, we vary k from 100 to 1400 in increments of 100 (14 values for k) and we vary p from -1 to $+1$ in increments of 0.1 (21 values for p). When p is -1, we give more weight to the factors with smaller singular values; when p is $+1$, the factors with larger singular values have more weight. Caron (2001) observes that most researchers use either $p = 0$ or $p = 1$; that is, they use either $\mathbf{U}_k$ or $\mathbf{U}_k \mathbf{\Sigma}_k$.

Let $\text{sim}_\text{f}(a, b) \in \Re$ be function similarity as measured by the cosine of vectors $\mathbf{a}$ and $\mathbf{b}$ in function space. Let $\text{sim}_\text{d}(a, b) \in \Re$ be domain similarity as measured by the cosine of vectors $\mathbf{a}$ and $\mathbf{b}$ in domain space. When a similarity measure combines both $\text{sim}_\text{d}(a, b)$ and $\text{sim}_\text{f}(a, b)$, there are four parameters to tune, k_d and p_d for domain space and k_f and p_f for function space.

For one space, it is feasible for us to explore all $14 \times 21 = 294$ combinations of parameter values, but two spaces have $294 \times 294 = 86,436$ combinations of values. To make the search tractable, we initialize the parameters to the middle of their ranges ($k_\text{f} = k_\text{d} = 700$ and $p_\text{f} = p_\text{d} = 0$) and then we alternate between tuning $\text{sim}_\text{d}(a, b)$ (i.e., k_d and p_d) while holding $\text{sim}_\text{f}(a, b)$ (i.e., k_f and p_f) fixed and tuning $\text{sim}_\text{f}(a, b)$ while holding $\text{sim}_\text{d}(a, b)$ fixed. We stop the search when there is no improvement in performance on the training data. In almost all cases, a local optimum is found in one pass; that is, after we have tuned the parameters once, there is no improvement when we try to tune them a second time. Thus we typically evaluate $294 \times 3 = 882$ parameter values ($\times 3$ because we tune one similarity, tune the other, and then try the first again to see if further improvement is possible).[11]

We could use a standard numerical optimization algorithm to tune the four parameters, but the algorithm we use here takes advantage of background knowledge about the optimization task. We know that small variations in the parameters make small changes in performance, so there is no need to make a very fine-grained search, and we know that $\text{sim}_\text{d}(a, b)$ and $\text{sim}_\text{f}(a, b)$ are relatively independent, so we can optimize them separately.

The rows in the matrices are based on terms in the WordNet *index.sense* file. In this file, all nouns are in their singular forms and all verbs are in their stem forms. To calculate $\text{sim}(a, b)$, we first look for exact matches for a and b in the terms that correspond to the rows of the given matrix (domain, function, or mono). If an exact match is found, then we use the corresponding row vector in the matrix. Otherwise, we look for alternate forms of the terms, using the *validForms* function in the WordNet::QueryData Perl interface to WordNet.[12] This automatically converts plural nouns to their singular forms and verbs to their stem forms. If none of the alternate forms is an exact match for a row in the matrix, we map the term to a zero vector of length k.

11. We use Perl Data Language (PDL) for searching for parameters, calculating cosines, and other operations on vectors and matrices. See http://pdl.perl.org/.
12. WordNet::QueryData is available at http://search.cpan.org/dist/WordNet-QueryData/.

3.7 Composing Similarities

Our approach to semantic relations and compositions is to combine the two similarities, $\mathrm{sim_d}(a, b)$ and $\mathrm{sim_f}(a, b)$, in various ways, depending on the task at hand or the syntax of the phrase at hand. In general, we want the combined similarity to be high when the component similarities are high, and we want the values of the component similarities to be balanced. To achieve balance, we use the geometric mean to combine similarities, instead of the arithmetic mean. The geometric mean is not suitable for negative numbers, and the cosine can be negative in some cases; hence we define the geometric mean as zero if any of the component similarities are negative:

$$\mathrm{geo}(x_1, x_2, \ldots, x_n) = \begin{cases} (x_1 x_2 \ldots x_n)^{1/n} & \text{if } x_i > 0 \text{ for all } i = 1, \ldots, n \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

3.8 Element-wise Multiplication

One of the most successful approaches to composition, so far, has been element-wise multiplication, $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$, where $c_i = a_i \cdot b_i$ (Mitchell & Lapata, 2008, 2010). This approach only makes sense when the elements in the vectors are not negative. When the elements in $\mathbf{a}$ and $\mathbf{b}$ are positive, relatively large values of a_i and b_i reinforce each other, resulting in a large value for c_i. This makes intuitive sense. But when a_i and b_i are both highly negative, c_i will be highly positive, although intuition says c_i should be highly negative. Mitchell and Lapata (2008, 2010) designed their word–context matrices to ensure that the vectors had no negative elements.

The values in the matrix $\mathbf{U}_k \boldsymbol{\Sigma}_k^p$ are typically about half positive and half negative. We use element-wise multiplication as a baseline in some of the following experiments. For a fair baseline, we cannot simply apply element-wise multiplication to row vectors in $\mathbf{U}_k \boldsymbol{\Sigma}_k^p$. One solution would be to use the PPMI matrix, $\mathbf{X}$, which has no negative elements, but this would not allow element-wise multiplication to take advantage of the smoothing effect of SVD. Our solution is to use row vectors from $\hat{\mathbf{X}} = \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^\mathsf{T}$. Although the PPMI matrix, $\mathbf{X}$, is sparse (see Table 5), $\hat{\mathbf{X}}$ and $\mathbf{U}_k \boldsymbol{\Sigma}_k^p$ have a density of 100%.

Let $\mathbf{a}'$ and $\mathbf{b}'$ be the vectors in $\hat{\mathbf{X}}$ that correspond to the terms a and b. These row vectors benefit from smoothing due to truncated SVD, but their elements are almost all positive. If there are any negative elements, we set them to zero. Let $\mathbf{c}' = \mathbf{a}' \odot \mathbf{b}'$. After we apply element-wise multiplication to the vectors, we then multiply by $\mathbf{V}_k \boldsymbol{\Sigma}_k^{p-1}$, so that the resulting vector $\mathbf{c} = \mathbf{c}' \mathbf{V}_k \boldsymbol{\Sigma}_k^{p-1}$ can be compared with other row vectors in the matrix $\mathbf{U}_k \boldsymbol{\Sigma}_k^p$:

$$\begin{aligned} \hat{\mathbf{X}}(\mathbf{V}_k \boldsymbol{\Sigma}_k^{p-1}) &= (\mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^\mathsf{T})(\mathbf{V}_k \boldsymbol{\Sigma}_k^{p-1}) && (12) \\ &= \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}_k^\mathsf{T} \mathbf{V}_k \boldsymbol{\Sigma}_k^{p-1} && (13) \\ &= \mathbf{U}_k \boldsymbol{\Sigma}_k \boldsymbol{\Sigma}_k^{p-1} && (14) \\ &= \mathbf{U}_k \boldsymbol{\Sigma}_k^p && (15) \end{aligned}$$

Note that, since $\mathbf{V}_k$ is column orthonormal, $\mathbf{V}_k^\mathsf{T} \mathbf{V}_k$ equals $\mathbf{I}_k$, the $k \times k$ identity matrix.

Similarly, if $\mathbf{a}$ is a row vector in $\mathbf{U}_k\mathbf{\Sigma}_k^p$, we can find its counterpart $\mathbf{a}'$ in $\hat{\mathbf{X}}$ by multiplying $\mathbf{a}$ with $\mathbf{\Sigma}_k^{1-p}\mathbf{V}_k^\mathsf{T}$:

$$(\mathbf{U}_k\mathbf{\Sigma}_k^p)(\mathbf{\Sigma}_k^{1-p}\mathbf{V}_k^\mathsf{T}) = \mathbf{U}_k\mathbf{\Sigma}_k^p\mathbf{\Sigma}_k^{1-p}\mathbf{V}_k^\mathsf{T} \tag{16}$$

$$= \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\mathsf{T} \tag{17}$$

$$= \hat{\mathbf{X}} \tag{18}$$

Let $\mathrm{nn}(\mathbf{x})$ (*nn* for *nonnegative*) be a function that converts negative elements in a vector $\mathbf{x}$ to zero:

$$\mathrm{nn}(\langle x_1, \ldots, x_n \rangle) = \langle y_1, \ldots, y_n \rangle \tag{19}$$

$$y_i = \begin{cases} x_i & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

Our version of element-wise multiplication may be expressed as follows:

$$\mathbf{c} = (\mathrm{nn}(\mathbf{a}\mathbf{\Sigma}_k^{1-p}\mathbf{V}_k^\mathsf{T}) \odot \mathrm{nn}(\mathbf{b}\mathbf{\Sigma}_k^{1-p}\mathbf{V}_k^\mathsf{T}))\, \mathbf{V}_k\mathbf{\Sigma}_k^{p-1} \tag{21}$$

Another way to deal with element-wise multiplication would be to use nonnegative matrix factorization (NMF) (Lee & Seung, 1999) instead of SVD. We have not yet found an implementation of NMF that scales to the matrix sizes that we have here (Table 5). In our past experiments with smaller matrices, SVD and NMF have similar performance.

4. Experiments with Varieties of Similarities

This section presents four sets of experiments. The first set of experiments presents a dual-space model of semantic relations and evaluates the model with multiple choice analogy questions from the SAT. The second set presents a model of semantic composition and evaluates it with multiple choice questions that are constructed from WordNet. The third set applies a dual-space model to the phrase similarity dataset of Mitchell and Lapata (2010). The final set uses three classes of word pairs from Chiarello et al. (1990) to test a hypothesis about the dual-space model, that domain space and function space capture the intuitive concepts of *association* and *similarity.*

4.1 Similarity of Relations

Here we evaluate the dual-space model applied to the task of measuring the similarity of semantic relations. We use a set of 374 multiple-choice analogy questions from the SAT college entrance exam (Turney, 2006b). Table 6 gives an example of one of the questions. The task is to select the choice word pair that is most analogous (most relationally similar) to the stem word pair.

Let $a : b$ represent the stem pair (e.g., *lull*:*trust*). We answer the SAT questions by selecting the choice pair $c\!:\!d$ that maximizes the relational similarity, $\mathrm{sim_r}(a\!:\!b, c\!:\!d)$, defined as follows:

Stem:		lull:trust
Choices:	(1)	balk:fortitude
	(2)	betray:loyalty
	(3)	cajole:compliance
	(4)	hinder:destination
	(5)	soothe:passion
Solution:	(3)	cajole:compliance

Table 6: An example of a question from the 374 SAT analogy questions. Lulling a person into trust is analogous to cajoling a person into compliance.

$$\mathrm{sim}_1(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim_f}(a, c), \mathrm{sim_f}(b, d)) \tag{22}$$

$$\mathrm{sim}_2(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim_d}(a, b), \mathrm{sim_d}(c, d)) \tag{23}$$

$$\mathrm{sim}_3(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim_d}(a, d), \mathrm{sim_d}(c, b)) \tag{24}$$

$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) = \begin{cases} \mathrm{sim}_1(a\!:\!b, c\!:\!d) & \text{if } \mathrm{sim}_2(a\!:\!b, c\!:\!d) \geq \mathrm{sim}_3(a\!:\!b, c\!:\!d) \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

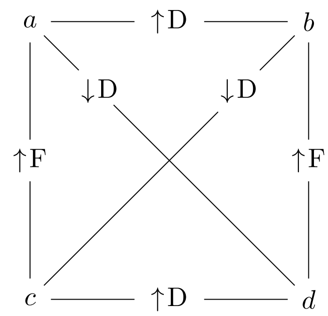
The intent of sim_1 is to measure the function similarity across the two pairs. The domain similarity *inside* the two pairs is measured by sim_2, whereas the domain similarity *across* the two pairs is given by sim_3. The relational similarity, $\mathrm{sim_r}$, is simply the function similarity, sim_1, subject to the constraint that the domain similarity inside pairs, sim_2, must not be less than the domain similarity across pairs, sim_3.

Figure 1 conveys the main ideas behind Equations 22 to 25. We want high function similarities (indicated by $\uparrow$ F) for $a : c$ and $b : d$, as measured by sim_1. We also prefer relatively high domain similarities (marked with $\uparrow$D) for $a\!:\!b$ and $c\!:\!d$ (measured by sim_2), in contrast to relatively low domain similarities ($\downarrow$D) for $a\!:\!d$ and $c\!:\!b$ (as given by sim_3).[13]

Using the example in Table 6, we see that lulling a person into trust is analogous to cajoling a person into compliance, since the functional role of *lull* is similar to the functional role of *cajole* (both involve manipulating a person) and the functional role of *trust* is similar to the functional role of *compliance* (both are states that a person can be in). This is captured by sim_1. The constraint $\mathrm{sim}_2(a : b, c : d) \geq \mathrm{sim}_3(a : b, c : d)$ implies that the domain similarities of *lull*:*trust* (the domain of confidence and loyalty) and *cajole*:*compliance* (the domain of obedience and conformity) should be greater than or equal to the domain similarities of *lull*:*compliance* and *cajole*:*trust*.

Analogy is a way of mapping knowledge from a source domain to a target domain (Gentner, 1983). If a in the source domain is mapped to c in the target domain, then a should play the same role in the source domain as c plays in the target domain. This is the theory behind sim_1. If a and b are in the source domain and c and d are in the target

13. We recently came across this same rectangular structure in Lepage and Shin-ichi's (1996) paper on morphological analogy (see their Figure 1). Although our algorithm and our task differ considerably from the algorithm and task of Lepage and Shin-ichi (1996), we have independently discovered the same underlying structure in analogical reasoning.



$\mathrm{sim_r}(a\!:\!b, c\!:\!d)$

relational similarity

Figure 1: A diagram of the reasoning behind Equations 22 to 25. $\uparrow$ F represents high function similarity, $\uparrow$ D means high domain similarity, and $\downarrow$ D indicates low domain similarity.

domain, then the internal domain similarity of a and b and the internal domain similarity of c and d should not be less than the cross-domain similarities. This motivates the constraint $\mathrm{sim}_2 \geq \mathrm{sim}_3$. Our definition is a natural expression of Gentner's (1983) theory of analogy.

Recall the four equations that we introduced in Section 2.2. We repeat these equations here for convenience:

$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) = \mathrm{sim_r}(b\!:\!a, d\!:\!c) \tag{26}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) = \mathrm{sim_r}(c\!:\!d, a\!:\!b) \tag{27}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) \neq \mathrm{sim_r}(a\!:\!b, d\!:\!c) \tag{28}$$
$$\mathrm{sim_r}(a\!:\!b, c\!:\!d) \neq \mathrm{sim_r}(a\!:\!d, c\!:\!b) \tag{29}$$

Inspection will show that the definition of relational similarity in Equation 25 satisfies the requirements of Equations 26, 27, 28, and 29. This can be understood by considering Figure 1. Equation 26 tells us that we can rotate Figure 1 about its vertical axis without altering the network of similarities, due to the symmetry of the figure. Equation 27 tells us that we can rotate Figure 1 about its horizontal axis without altering the network of similarities.

On the other hand, we cannot swap c and d while holding a and b fixed, because this would change both the $\uparrow$ F and $\downarrow$ D links (although it would not change the $\uparrow$ D links). In other words, sim_1 and sim_3 would be changed, although sim_2 would not be affected. Therefore Equation 28 is satisfied.

Also, we cannot swap b and d while holding a and c fixed, because this would change the $\uparrow$ D and $\downarrow$ D links (although it would not change the $\uparrow$ F links). In other words, sim_2 and sim_3 would be changed, although sim_1 would not be affected. Therefore Equation 29 is satisfied. We can see that sim_1 by itself would violate Equation 29, due to the symmetry of cosines, $\text{sim}_\text{f}(b, d) = \text{sim}_\text{f}(d, b)$. The constraint $\text{sim}_2(a:b, c:d) \geq \text{sim}_3(a:b, c:d)$ breaks this symmetry.

Another way to break the symmetry, so that Equation 29 is satisfied, would be to use a similarity measure that is inherently asymmetric, such as skew divergence. In Equation 25, the symmetry is broken in a natural way by considering how domain and function similarity apply to analogies, so there is no need to introduce an inherently asymmetric measure. Also, note that the symmetries of Equations 26 and 27 are desirable; we do not wish to break these symmetries.

It would have been reasonable to include $\text{sim}_\text{d}(a, c)$ and $\text{sim}_\text{d}(b, d)$ in sim_3, but we decided to leave them out. It seems to us that the function similarities $\text{sim}_\text{f}(a, c)$ and $\text{sim}_\text{f}(b, d)$, which should have high values in a good analogy, might cause $\text{sim}_\text{d}(a, c)$ and $\text{sim}_\text{d}(b, d)$ to be relatively high, even though they cross domains. If people observe a certain kind of abstract function similarity frequently, that function similarity might become a popular topic for discussion, which could result in a high domain similarity.

For example, *carpenter*:*wood* is analogous to *mason*:*stone*. The domain of *carpenter*:*wood* is carpentry and the domain of *mason*:*stone* is masonry. The functional role of *carpenter* is similar to the functional role of *mason*, in that both are artisans. Although *carpenter* and *mason* belong to different domains, their high degree of abstract function similarity may result in discussions that mention them together, such as discussions about specialized trades, skilled manual labour, the construction industry, and workplace injuries. In other words, high function similarity between two words may cause a rise in their domain similarity. Therefore we did not include $\text{sim}_\text{d}(a, c)$ and $\text{sim}_\text{d}(b, d)$ in sim_3.

When all five choices for a SAT question have a relational similarity of zero, we skip the question. We use ten-fold cross-validation to set the parameters for the SAT questions. The same parameter values are selected in nine of the ten folds, $k_\text{d} = 800$, $p_\text{d} = -0.1$, $k_\text{f} = 300$, and $p_\text{f} = 0.5$. After the parameters are determined, all 374 SAT questions can be answered in a few seconds. Equation 25 correctly answers 191 of the questions, skips 2 questions, and incorrectly answers 181 questions, achieving an accuracy of 51.1%.

4.1.1 Comparison with Past Work

For comparison, the average score for senior highschool students applying to US universities is 57.0%. The ACL Wiki lists many past results with the 374 SAT questions.[14] Table 7 shows the top ten results at the time of writing. In this table, *dual-space* refers to the dual-space model using Equation 25. Four of the past results achieved an accuracy of 51.1% or higher. All four used holistic approaches and hence are not able to address the issue of linguistic creativity. The best previous algorithm attains an accuracy of 56.1% (210 correct, 4 skipped, 160 incorrect) (Turney, 2006b). The difference between 51.1% and 56.1% is not statistically significant at the 95% confidence level, according to Fisher's Exact Test.

The majority of the algorithms in Table 7 are unsupervised, but Dual-Space, PairClass (Turney, 2008b), and BagPack (Herdağdelen & Baroni, 2009) use limited supervision. PairClass and BagPack answer a given SAT question by learning a binary classification model that is specific to the given question. The training set for a given question consists of one

14. See http://aclweb.org/aclwiki/index.php?title=SAT_Analogy_Questions.

Algorithm	Reference	Accuracy	95% confidence
LSA+Predication	Mangalath et al. (2004)	42.0	37.2–47.4
KNOW-BEST	Veale (2004)	43.0	38.0–48.2
k-means	Biçici and Yuret (2006)	44.0	39.0–49.3
BagPack	Herdağdelen and Baroni (2009)	44.1	39.0–49.3
VSM	Turney and Littman (2005)	47.1	42.2–52.5
Dual-Space		51.1	46.1–56.5
BMI	Bollegala et al. (2009)	51.1	46.1–56.5
PairClass	Turney (2008b)	52.1	46.9–57.3
PERT	Turney (2006a)	53.5	48.5–58.9
LRA	Turney (2006b)	56.1	51.0–61.2
Human	Average US college applicant	57.0	52.0–62.3

Table 7: The top ten results with the 374 SAT questions, from the ACL Wiki. The 95% confidence intervals are calculated using the Binomial Exact Test.

positive training example, the stem pair for the question, and ten randomly selected pairs as (assumed) negative training examples. The induced binary classifier is used to assign probabilities to the five choices and the most probable choice is the guess. Dual-Space uses the training set only to tune four numerical parameters. These three algorithms are best described as *weakly* supervised.

4.1.2 Sensitivity to Parameters

To see how sensitive the dual-space model is to the values of the parameters, we perform two exhaustive grid searches, one with a coarse, wide grid and another with a fine, narrow grid. For each point in the grids, we evaluate the dual-space model using the whole set of 374 SAT questions. The narrow grid search is centred on the parameter values that were selected in nine of the ten folds in the previous experiment, $k_\mathrm{d} = 800$, $p_\mathrm{d} = -0.1$, $k_\mathrm{f} = 300$, and $p_\mathrm{f} = 0.5$. Both searches evaluate 5 values for each parameter, yielding a total of $5^4 = 625$ parameter settings. Table 8 shows the values that were explored in the two grid searches and Table 9 presents the minimum, maximum, average, and standard deviation of the accuracy for the two searches.

Grid	Parameter	Values				
Coarse	k_d	100	425	750	1075	1400
	p_d	-1.0	-0.5	0.0	0.5	1.0
	k_f	100	425	750	1075	1400
	p_f	-1.0	-0.5	0.0	0.5	1.0
Fine	k_d	600	700	800	900	1000
	p_d	-0.3	-0.2	-0.1	0.0	0.1
	k_f	100	200	300	400	500
	p_f	0.3	0.4	0.5	0.6	0.7

Table 8: The range of parameter values for the two grid searches.

	Accuracy			
Grid	Minimum	Maximum	Average	Standard deviation
Coarse	31.0	48.7	40.7	4.1
Fine	42.5	51.6	47.3	2.0

Table 9: The sensitivity of the dual-space model to the parameter settings.

The accuracy attained by the heuristic search (described in Section 3.6) with ten-fold cross-validation, 51.1% (Table 7), is near the best accuracy of the fine grid search using the whole set of 374 SAT questions, 51.6% (Table 9). This is evidence that the heuristic search is effective. Accuracy with the coarse search varies from 31.0% to 48.7%, which demonstrates the importance of tuning the parameters. On the other hand, accuracy with the fine search spans a narrower range and has a lower standard deviation, which suggests that the dual-space model is not overly sensitive to relatively small variations in the parameter values; that is, the parameters are reasonably stable. (That nine of the ten folds in cross-validation select the same parameters is further evidence of stability.)

4.1.3 Parts of Speech

Since domain space is based on nouns and function space is based on verbs, it is interesting to know how the performance of the dual-space model varies with different parts of speech. To answer this, we manually labeled all 374 SAT questions with part-of-speech labels. The labels for a single pair can be ambiguous, but the labels become unambiguous in the context of the whole question. For example, *lull*:*trust* could be *noun*:*verb*, but in the context of Table 6, it must be *verb*:*noun*.

Table 10 splits out the results for the various parts of speech. None of the differences in this table are statistically significant at the 95% confidence level, according to Fisher's Exact Test. A larger and more varied set of questions will be needed to determine how part of speech affects the dual-space model.

Parts of speech	Right	Accuracy	Wrong	Skipped	Total
noun:noun	97	50.8	93	1	191
noun:adjective or adjective:noun	35	53.0	31	0	66
noun:verb or verb:noun	27	49.1	28	0	55
adjective:adjective	9	37.5	15	0	24
verb:adjective or adjective:verb	12	60.0	7	1	20
verb:verb	11	64.7	6	0	17
verb:adverb or adverb:verb	0	0.0	1	0	1
all	191	51.1	181	2	374

Table 10: Performance of the dual-space model with various parts of speech.

4.1.4 Order Sensitivity

It seems that function space is doing most of the work in Equation 25. If we use sim_1 alone, dropping the constraint that $\text{sim}_2 \geq \text{sim}_3$, then accuracy drops from 51.1% to 50.8%. This drop is not statistically significant. We hypothesize that the small drop is due to the design of the SAT test, which is primarily intended to test a student's understanding of functional roles, not domains.

To verify this hypothesis, we reformulated the SAT questions so that they would test both function and domain comprehension. The method is to first expand each choice pair $c\!:\!d$ by including the stem pair $a\!:\!b$, resulting in the full explicit analogy $a\!:\!b\!::\!c\!:\!d$. For each expanded choice, $a : b :: c : d$, we then generate another choice, $a : d :: c : b$. Table 11 shows the reformulation of Table 6. Due to symmetry, sim_1 must assign the same similarity to both $a\!:\!b\!::\!c\!:\!d$ and $a\!:\!d\!::\!c\!:\!b$. This new ten-choice test evaluates both function and domain similarities.

Choices:	(1)	lull:trust::balk:fortitude
	(2)	lull:fortitude::balk:trust
	(3)	lull:loyalty::betray:trust
	(4)	lull:trust::betray:loyalty
	(5)	lull:compliance::cajole:trust
	(6)	lull:trust::cajole:compliance
	(7)	lull:destination::hinder:trust
	(8)	lull:trust::hinder:destination
	(9)	lull:trust::soothe:passion
	(10)	lull:passion::soothe:trust
Solution:	(6)	lull:trust::cajole:compliance

Table 11: An expanded SAT question, designed to test both function and domain comprehension. Choices (5) and (6) have the same similarity according to sim_1.

The task with the expanded ten-choice SAT questions is the same as with the original five-choice questions, to select the best analogy. The solution in Table 11 is the same as the solution in Table 6, except that the stem pair is explicit in Table 11. The only signficant change is that five new distractors have been added to the choices. We answer the ten-choice questions by selecting the choice $a\!:\!b\!::\!c\!:\!d$ that maximizes $\mathrm{sim_r}(a\!:\!b, c\!:\!d)$.

On the ten-choice reformulated SAT test, $\mathrm{sim_r}$ (Equation 25) attains an accuracy of 47.9%, whereas sim_1 alone (Equation 22) only achieves 27.5%. The difference is statistically significant at the 95% confidence level, according to Fisher's Exact Test. This more stringent test supports the claim that function similarity is insufficient by itself.

As a further test of the value of two separate spaces, we use a single space for both $\mathrm{sim_d}$ and $\mathrm{sim_f}$ in Equation 25. The model still has four parameters it can tune, k_d, p_d, k_f, and p_f, but the same matrix is used for both similarities. The best result is an accuracy of 40.4% on the ten-question reformulated SAT test, using function space for both $\mathrm{sim_d}$ and $\mathrm{sim_f}$. This is significantly below the 47.9% accuracy of the dual-space model when $\mathrm{sim_d}$ is based on domain space and $\mathrm{sim_f}$ is based on function space (95% confidence level, Fisher's Exact Test).

Table 12 summarizes the results. In the cases where the matrix for $\mathrm{sim_d}$ is *not used*, the model is based on sim_1 alone (Equation 22). In all other cases, the model is based on $\mathrm{sim_r}$ (Equation 25). For both the five-choice and ten-choice SAT questions, the original dual-space model is more accurate than any of the modified models. The *Significant* column indicates whether the accuracy of a modified model is significantly less than the original dual-space model (95% confidence level, Fisher's Exact Test). The more difficult ten-choice questions clearly show the value of two distinct spaces.

Algorithm	Accuracy	Significant	Questions	Matrix for $\mathrm{sim_d}$	Matrix for $\mathrm{sim_f}$
dual-space	51.1		five-choice	domain space	function space
modified dual-space	47.3	no	five-choice	function space	function space
modified dual-space	43.6	yes	five-choice	mono space	mono space
modified dual-space	37.7	yes	five-choice	domain space	domain space
modified dual-space	50.8	no	five-choice	not used	function space
modified dual-space	41.7	yes	five-choice	not used	mono space
modified dual-space	35.8	yes	five-choice	not used	domain space
dual-space	47.9		ten-choice	domain space	function space
modified dual-space	40.4	yes	ten-choice	function space	function space
modified dual-space	38.2	yes	ten-choice	mono space	mono space
modified dual-space	34.8	yes	ten-choice	domain space	domain space
modified dual-space	27.5	yes	ten-choice	not used	function space
modified dual-space	25.1	yes	ten-choice	not used	mono space
modified dual-space	14.4	yes	ten-choice	not used	domain space

Table 12: Accuracy with the original five-choice questions and the reformulated ten-choice questions. In the modified models, we intentionally use the wrong matrix (or no matrix) for $\mathrm{sim_d}$ or $\mathrm{sim_f}$. The modified models show that accuracy decreases when only one space is used.

4.1.5 Summary

The dual-space model performs as well as the current state-of-the-art holistic model and addresses the issue of linguistic creativity. The results with the reformulated SAT questions support the claim that there is value in having two separate spaces.

As we mentioned in Section 2.2, the task of classifying word pairs according to their semantic relations (Rosario & Hearst, 2001; Rosario et al., 2002; Nastase & Szpakowicz, 2003) is closely connected to the problem of measuring relational similarity. Turney (2006b) applied a measure of relational similarity to relation classification by using cosine similarity as a measure of nearness in a nearest neighbour supervised learning algorithm. The dual-space model (Equation 25) is also suitable for relation classification with a nearest neighbour algorithm.

4.2 Similarity of Compositions

In this second set of experiments, we apply the dual-space model to noun-modifier compositions. Given vectors for *dog*, *house*, and *kennel*, we would like to be able to recognize that *dog house* and *kennel* are synonymous. We compare the dual-space model to the holistic approach, vector addition, and element-wise multiplication. The approaches are evaluated using multiple-choice questions that are automatically generated from WordNet, using the WordNet::QueryData Perl interface to WordNet. Table 13 gives an example of one of the noun-modifier questions.

Stem:		dog house
Choices:	(1)	kennel
	(2)	dog
	(3)	house
	(4)	canine
	(5)	dwelling
	(6)	effect
	(7)	largeness
Solution:	(1)	kennel

Table 13: An example of a multiple-choice noun-modifier composition question.

In these questions, the stem is a bigram and the choices are unigrams. Choice (1) is the correct answer, (2) is the modifier, and (3) is the head noun. Choice (4) is a synonym or hypernym of the modifier and (5) is a synonym or hypernym of the head noun. If no synonyms or hypernyms can be found, a noun is randomly chosen. The last two choices, (6) and (7), are randomly selected nouns. Choices (2) and (4) can be either nouns or adjectives, but the other choices must be nouns.

The stem bigram and the choice unigrams must have corresponding rows in function space (the space with the least number of rows). The stem bigram must have a noun sense in WordNet (it may also have senses for other parts of speech). The solution unigram, (1), must be a member of the synset (synonym set) for the first noun sense of the stem bigram (the most frequent or dominant sense of the bigram, when the bigram is used as a noun), and it cannot be simply the hyphenation (*dog-house*) or concatenation (*doghouse*) of the stem bigram.

These requirements result in a total of 2180 seven-choice questions, which we randomly split into 680 for training (parameter tuning) and 1500 for testing.[15] The questions are deliberately designed to be difficult. In particular, all of the approaches are strongly attracted to choices (2) and (3). Furthermore, we did not attempt to ensure that the stem bigrams are compositional; some of them may be idiomatic expressions that no compositional approach could possibly get right. We did not want to bias the questions by imposing theories about distinguishing compositions and idioms in their construction.

Let ab represent a noun-modifier bigram (*dog house*) and let c represent a unigram (*kennel*). We answer the multiple-choice questions by selecting the unigram that maximizes the compositional similarity, $\mathrm{sim_c}(ab, c)$, defined as follows:

$$\mathrm{sim}_1(ab, c) = \mathrm{geo}(\mathrm{sim_d}(a, c), \mathrm{sim_d}(b, c), \mathrm{sim_f}(b, c)) \tag{30}$$

$$\mathrm{sim_c}(ab, c) = \begin{cases} \mathrm{sim}_1(ab, c) & \text{if } a \neq c \text{ and } b \neq c \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

15. The questions are available as an online appendix at http://jair.org/.

Equations 30 and 31 are illustrated in Figure 2.

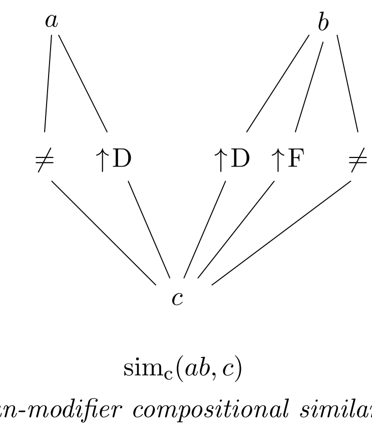


Figure 2: A diagram of Equations 30 and 31.

The thinking behind sim_1 is that c (*kennel*) should have high domain similarity with both the modifier a (*dog*) and the head noun b (*house*); furthermore, the function of the bigram ab (*dog house*) is determined by the head noun b (*house*), so the head noun should have high function similarity with c (*kennel*). We add the constraints $a \neq c$ and $b \neq c$ because sim_1 by itself tends to have high values for $\text{sim}_1(ab, a)$ and $\text{sim}_1(ab, b)$.[16] It seems plausible that humans use constraints like this: We reason that *dog house* cannot mean the same thing as *house*, because then the extra word *dog* in *dog house* would serve no purpose; it would be meaningless noise.[17]

The constraints $a \neq c$ and $b \neq c$ could be expressed in terms of similarities, such as $\text{sim}_\text{d}(a, c) < t$ and $\text{sim}_\text{d}(b, c) < t$, where t is a high threshold (e.g., $t = 0.9$), but this would add another parameter to the model. We decided to keep the model relatively simple.

When all seven choices for a noun-modifier question have a compositional similarity of zero, we skip the question. On the training set, the best parameter settings are $k_\text{d} = 800$, $p_\text{d} = 0.3$, $k_\text{f} = 100$, and $p_\text{f} = 0.6$. On the testing set, Equation 31 correctly answers 874 questions, skips 22 questions, and incorrectly answers 604, yielding an accuracy of 58.3%.

4.2.1 Comparison with Other Approaches

Mitchell and Lapata (2010) compared many different approaches to semantic composition in their experiments, but they only considered one task (the task we examine in Section 4.3). In this paper, we have chosen to compare a smaller number of approaches on a larger number of tasks. We include element-wise multiplication in these experiments, because this approach had the best performance in Mitchell and Lapata's (2010) experiments. Vector

16. In spite of these constraints, it is still worthwhile to include the head noun and the modifier as distractors in the multiple-choice questions, because it enables us to experimentally evaluate the impact of these distractors on the various algorithms when the constraints are removed (see Table 15). Also, future users of this dataset may find a way to avoid these distractors without explicit constraints.
17. In philosophy of language, Grice (1989) argued that proper interpretation of language requires us to charitably assume that speakers generally do not insert random words into their speech.

addition is included due to its historical importance and its simplicity. Although Mitchell and Lapata (2010) found that weighted addition was better than unweighted addition, we do not include weighted addition in our experiments, because it did not perform as well as element-wise multiplication in Mitchell and Lapata's (2010) experiments. We include the holistic model as a noncompositional baseline.

Table 14 compares the dual-space model with the holistic model, element-wise multiplication, and vector addition. For the latter three models, we try all three spaces.

Algorithm	Space	Accuracy
dual-space	domain and function	58.3
holistic	mono	81.6
holistic	domain	79.1
holistic	function	67.5
multiplication	mono	55.7
multiplication	domain	57.5
multiplication	function	46.3
addition	mono	48.3
addition	domain	50.1
addition	function	39.8

Table 14: Results for the noun-modifier questions.

In this table, *dual-space* refers to the dual-space model using Equation 31. In the holistic model, ab is represented by its corresponding row vector in the given space. Recall from Section 3.1 that, in Step 1.1, the rows in the matrices correspond to n-grams in WordNet, where n may be greater than one. Thus, for example, *dog house* has a corresponding row vector in all three of the spaces. The holistic model simply uses this row vector as the representation of *dog house*. For element-wise multiplication, ab is represented using Equation 21. With the vector addition model, ab is represented by $\mathbf{a}+\mathbf{b}$, where the vectors are normalized to unit length before they are added. All four models use the constraints $a \neq c$ and $b \neq c$. All four models use the training data for parameter tuning.

The difference between the dual-space model (58.3%) and the best variation of element-wise multiplication (57.5%) is not statistically significant at the 95% confidence level, according to Fisher's Exact Test. However, the difference between the dual-space model (58.3%) and the best variation of vector addition (50.1%) is significant.

4.2.2 Limitations of the Holistic Approach

For all three spaces, the holistic model is significantly better than all other models, but its inability to address the issue of linguistic creativity is a major limitation. The 2180 multiple-choice questions that we have used in these experiments were intentionally constructed with the requirement that the stem bigram must have a corresponding row in function space (see above). This was done so that we could use the holistic model as a baseline; however, it gives the misleading impression that the holistic model is a serious competitor with the compositional approaches. By design, Table 14 shows what the holistic model can achieve under ideal (but unrealistic) conditions.

Mitchell and Lapata's (2010) dataset, used in the experiments in Section 4.3, illustrates the limitations of the holistic model. The dataset consists of 324 distinct pairs of bigrams, composed of 216 distinct bigrams. Of the 216 bigrams, 28 (13%) occur in WordNet. Of the 324 pairs of bigrams, 13 (4%) contain bigrams that both occur in WordNet. Given the matrices we use here (with rows based on WordNet), the holistic approach would be reduced to random guessing for 96% of the pairs in Mitchell and Lapata's (2010) dataset.

It might be argued that the failure of the holistic approach with Mitchell and Lapata's (2010) dataset is due to our decision to base the rows of the matrices on terms from WordNet. However, suppose we attempt to build a holistic model for all frequent bigrams. The Web 1T 5-gram corpus (Brants & Franz, 2006) includes a list of all bigrams that appeared 40 or more times in a terabyte of text, a total of 314,843,401 bigrams. Using a compositional approach, the matrices we use here can represent the majority of these bigrams. On the other hand, the holistic approach would require a matrix with 314,843,401 rows, which is considerably beyond the current state of the art.

One possibility is to build a matrix for the holistic approach as needed, for a given input set of n-grams, instead of building a large, static, multipurpose matrix. There are two problems with this idea. First, it is slow. Turney (2006b) used this approach for the SAT analogy questions, but it required nine days to run, whereas the dual-space model can process the SAT questions in a few seconds, given a static, multipurpose matrix. Second, it requires a large corpus, and the corpus size must grow exponentially with n, the length of the phrases. Longer phrases are more rare, so larger corpora are needed to gather sufficient data to model the phrases. Larger corpora also result in longer processing times.

For a given application, it may be wise to have a predefined list of bigrams with holistic representations, but it would not be wise to expect this list to be sufficient to cover most bigrams that would be seen in practice. The creativity of human language use *requires* compositional models (Chomsky, 1975; Fodor & Lepore, 2002). Although the holistic model is included as a baseline in the experiments, it is not a *competitor* for the other models; it can only *supplement* the other models.

4.2.3 Impact of Constraints

If we use sim_1 alone (Equation 30), dropping the constraints $a \neq c$ and $b \neq c$, then accuracy drops signficantly, from 58.3% to 13.7%. However, all of the models benefit greatly from these constraints. In Table 15, we take the best variation of each model from Table 14 and look at what happens when the constraints are dropped.

		Accuracy		
Algorithm	Space	constraints	no constraints	difference
dual-space	domain and function	58.3	13.7	-44.6
holistic	mono	81.6	49.6	-32.0
multiplication	domain	57.5	8.2	-49.3
addition	domain	50.1	2.5	-47.6

Table 15: The impact of the constraints, $a \neq c$ and $b \neq c$, on accuracy.

4.2.4 Element-wise Multiplication

In Section 3.8, we argued that $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$ is not suitable for row vectors in the matrix $\mathbf{U}_k \mathbf{\Sigma}_k^p$ and we suggested Equation 21 as an alternative. When we use $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$ with domain space, instead of Equation 21, performance drops significantly, from 57.5% to 21.5%.

4.2.5 Impact of Idioms

Some of the gap between the holistic model and the other models may be due to idiomatic bigrams in the testing questions. One of the most successful approaches to determining whether a multiword expression (MWE) is compositional or noncompositional (idiomatic) is to compare its holistic vector representation with its compositional vector representation (for example, a high cosine between the two vectors suggests that the MWE is compositional, not idiomatic) (Biemann & Giesbrecht, 2011; Johannsen, Alonso, Rishøj, & Søgaard, 2011). However, this approach is not suitable here, because we do not want to *assume* that the gap is entirely due to idiomatic bigrams; instead, we would like to *estimate* how much of the gap is due to idiomatic bigrams.

WordNet contains some clues that we can use as indicators that a bigram might be less compositional than most bigrams (allowing that compositionality is a matter of degree). One clue is whether the WordNet gloss of the bigram contains either the head noun or the modifier. For example, the gloss of *dog house* is *outbuilding that serves as a shelter for a dog*, which contains the modifier, *dog*. This suggests that *dog house* may be compositional.

We classified each of the 1500 testing set questions as *head* (the first five characters in the *head* noun of the bigram match the first five characters in a word in the bigram's gloss), *modifier* (the first five characters in the *modifier* of the bigram match the first five characters in a word in the bigram's gloss), *both* (*both* the head and the modifier match), or *neither* (*neither* the head nor the modifier match). The four classes are approximately equally distributed in the testing questions (424 *head*, 302 *modifier*, 330 *both*, and 444 *neither*). We match on the first five characters to allow for cases like *brain surgeon*, which has the gloss *someone who does surgery on the nervous system (especially the brain)*. This bigram is classified as *both*, because the first five characters of *surgeon* match the first five characters of *surgery*.

Table 16 shows how the accuracy of the models varies over the four classes of questions. For the three compositional models (dual-space, multiplication, addition), the *neither* class is significantly less accurate than the other three classes (Fisher's Exact Test, 95% confidence), but the difference is not significant for the holistic model. For the three compositional models, the *neither* class is 17% to 20% less accurate than the other classes. This supports the view that a significant fraction of the wrong answers of the compositional models are due to noncompositional bigrams.

Another clue for compositionality in WordNet is whether the head noun is a hypernym of the bigram. For example, *surgeon* is a hypernym of *brain surgeon*. We classified each of the 1500 testing set questions as *hyper* (the head noun is a member of the synset of the immediate hypernym for the first noun sense of the bigram; we do not look further up in the hypernym hierarchy and we do not look at other senses of the bigram) or *not* (not *hyper*). In the testing set, 621 questions are *hyper* and 879 are *not*.

Algorithm	Space	Accuracy				
		both	head	modifier	neither	all
dual-space	domain and function	63.0	63.0	64.6	45.9	58.3
holistic	mono	82.7	83.7	82.1	78.4	81.6
multiplication	domain	61.8	63.7	62.9	44.8	57.5
addition	domain	53.6	56.8	56.3	36.7	50.1

Table 16: The variation of accuracy for different classes of bigram glosses.

Table 17 gives the accuracy of the models for each of the classes. This table has the same general pattern as Table 16. The three compositional models have significantly lower accuracy for the *not* class, with decreases from 6% to 8%. There is no significant difference for the holistic model.

Algorithm	Space	Accuracy		
		hyper	not	all
dual-space	domain and function	62.0	55.6	58.3
holistic	mono	81.0	82.0	81.6
multiplication	domain	61.8	54.5	57.5
addition	domain	54.8	46.8	50.1

Table 17: The variation of accuracy for different classes of bigram hypernyms.

4.2.6 Order Sensitivity

Note that vector addition and element-wise multiplication lack order sensitivity, but Equation 31 is sensitive to order, $\mathrm{sim_c}(ab, c) \neq \mathrm{sim_c}(ba, c)$. We can see the impact of this by reformulating the noun-modifier questions so that they test for order-sensitivity. First we expand each choice unigram c by including the stem bigram ab, resulting in the explicit comparison $ab \sim c$. For each expanded choice, $ab \sim c$, we then generate another choice, $ba \sim c$. This increases the number of choices from seven to fourteen. Due to symmetry, vector addition and element-wise multiplication must assign the same similarity to both $ab \sim c$ and $ba \sim c$.

Table 18 compares the dual-space model with element-wise multiplication and vector addition, using the reformulated fourteen-choice noun-modifier questions. The holistic model is not included in this table because there are no rows in the matrices for the reversed ba bigrams (which may be seen as another illustration of the limits of the holistic model). On this stricter test, the dual-space model is significantly more accurate than both element-wise multiplication and vector addition (Fisher's Exact Test, 95% confidence).

For the dual-space model to perform well with the fourteen-choice questions, we need both $\mathrm{sim_d}$ and $\mathrm{sim_f}$. If we drop $\mathrm{sim_d}$ from Equation 31 (*function alone* in Table 18), then we are ignoring the modifier and only paying attention to the head noun. Accuracy drops from 41.5% down to 25.7%. If we drop $\mathrm{sim_f}$ from Equation 31 (*domain alone* in Table 18), then the equation becomes symmetrical, so the same similarity is assigned to both $ab \sim c$ and

Algorithm	Space	Accuracy
dual-space	domain and function	41.5
multiplication	domain	27.4
modified dual-space	function alone	25.7
modified dual-space	domain alone	25.7
addition	domain	22.5

Table 18: Results for the reformulated fourteen-choice noun-modifier questions.

$ba \sim c$. Accuracy drops from 41.5% down to 25.7%.[18] The dual-space model is significantly more accurate than either of these modified dual-space models (Fisher's Exact Test, 95% confidence).

4.2.7 Summary

With the reformulated fourteen-choice noun-modifier questions (Table 18), the dual-space is significantly better than element-wise multiplication and vector addition. With the original seven-choice questions (Table 14), the difference is not as large, because these questions do not test for order. Unlike element-wise multiplication and vector addition, the dual-space model addresses the issue of order sensitivity. Unlike the holistic model, the dual-space addresses the issue of linguistic creativity.

4.3 Similarity of Phrases

In this subsection, we apply the dual-space model to measuring the similarity of phrases, using Mitchell and Lapata's (2010) dataset of human similarity ratings for pairs of phrases. The dataset includes three types of phrases, adjective-noun, noun-noun, and verb-object. There are 108 pairs of each type ($108 \times 3 = 324$ pairs of phrases). Each pair of phrases was rated by 18 human subjects. The ratings use a 7 point scale, in which 1 signifies the lowest degree of similarity and 7 signifies the highest degree. Table 19 gives some examples.

Let ab represent the first phrase in a pair of phrases (*environment secretary*) and let cd represent the second phrase (*defence minister*). We rate the similarity of the phrase pairs by $\text{sim}_\text{p}(ab, cd)$, defined as follows:

$$\text{sim}_\text{p}(ab, cd) = \text{geo}(\text{sim}_\text{d}(a, c), \text{sim}_\text{d}(b, d), \text{sim}_\text{f}(a, c), \text{sim}_\text{f}(b, d)) \tag{32}$$

This equation is based on the instructions to the human participants (Mitchell & Lapata, 2010, Appendix B), which imply that both function and domain similarity must be high for a phrase pair to get a high similarity rating. Figure 3 illustrates the reasoning behind this equation. We want high domain and function similarities between the corresponding components of the phrases ab and cd.

18. It is only a coincidence that both modified dual-space models have an accuracy of 25.7% on the fourteen-choice questions. Although their aggregate accuracy is the same, on individual questions, the two models typically select different choices.

Participant	Phrase type	Group	Phrase pair	Similarity
114	adjective-noun	2	certain circumstance ∼ particular case	6
114	adjective-noun	2	large number ∼ great majority	4
114	adjective-noun	2	further evidence ∼ low cost	2
109	noun-noun	0	environment secretary ∼ defence minister	6
109	noun-noun	0	action programme ∼ development plan	4
109	noun-noun	0	city centre ∼ research work	1
111	verb-object	2	lift hand ∼ raise head	7
111	verb-object	2	satisfy demand ∼ emphasise need	4
111	verb-object	2	like people ∼ increase number	1

Table 19: Examples of phrase pair similarity ratings from Mitchell and Lapata's (2010) dataset. Similarity ratings vary from 1 (lowest) to 7 (highest).

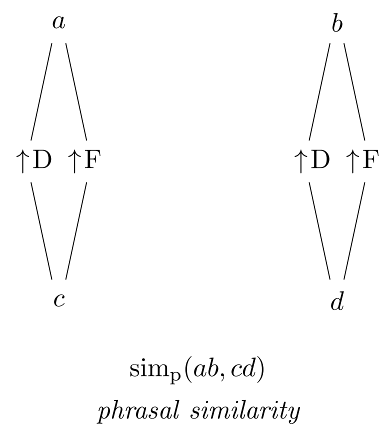


Figure 3: A diagram of Equation 32.

4.3.1 Experimental Setup

Mitchell and Lapata (2010) divided their dataset into a development set (for tuning parameters) and an evaluation set (for testing the tuned models). The development set has 6 ratings for each phrase pair and the evaluation set has 12 ratings for each phrase pair. The development and evaluation sets contain the same phrase pairs, but with judgments by different participants. Thus there are $6 \times 324 = 1,944$ rated phrase pairs in the development set and $12 \times 324 = 3,888$ ratings in the evaluation set.[19]

For a more challenging evaluation, we divide the dataset by phrase pairs rather than by participants. Our development set has 108 phrase pairs with 18 ratings each and the evaluation set has 216 phrase pairs with 18 ratings each. For each of the three phrase types, we randomly select 36 phrase pairs for the development set ($3 \times 36 = 108$ phrase pairs) and

19. The information in this paragraph is based on Section 4.3 of the paper by Mitchell and Lapata (2010) and personal communication with Jeff Mitchell in June, 2010.

72 for the evaluation set ($3 \times 72 = 216$ phrase pairs). Thus there are $18 \times 108 = 1,944$ ratings in the development set and $18 \times 216 = 3,888$ in the evaluation set.

Mitchell and Lapata (2010) use Spearman's rank correlation coefficient (Spearman's rho) to evaluate the performance of various vector composition algorithms on the task of emulating the human similarity ratings. For a given phrase type, the 108 phrase pairs are divided into 3 groups of 36 pairs each. For each group in the evaluation set, 12 people gave similarity ratings to the pairs in the given group. Each group of 36 pairs was given to a different group of 12 people. The score of an algorithm for a given phrase type is the average of three rho values, one rho for each of the three groups. With 12 people rating 36 pairs in a group, there are $12 \times 36 = 432$ ratings. These human ratings are represented by a vector of 432 numbers. An algorithm only generates one rating for each pair in a group, yielding 36 numbers. To make the algorithm's ratings comparable to the human ratings, the algorithm's ratings are duplicated 12 times, yielding a vector of 432 numbers. Spearman's rho is then calculated with these two vectors of 432 ratings. For 3 phrase types with 3 rho values each and 432 ratings per rho value, we have 3,888 ratings.[20]

We believe that this evaluation method underestimates the performance of the algorithms. Combining ratings from different people into one vector of 432 numbers does not allow the correlation to adapt to different biases. If one person gives consistently low ratings and another person gives consistently high ratings, but both people have the same ranking, and this ranking matches the algorithm's ranking, then the algorithm should get a high score. For a more fair evaluation, we score an algorithm by calculating one rho value for each human participant for the given phrase type, and then we calculate the average of the rho values for all of the participants.

For a given phrase type, the 108 phrase pairs are divided into 3 groups of 36 pairs each. For the development set, we randomly select 12 phrase pairs from each of the 3 groups ($3 \times 12 = 36$ phrase pairs per phrase type). This leaves 24 phrase pairs in each of the 3 groups for the evaluation set ($3 \times 24 = 72$ phrase pairs per phrase type). Each human participant's ratings are represented by a vector of 24 numbers. An algorithm's ratings are also represented by a vector of 24 numbers. A rho value is calculated with these two vectors of 24 numbers as input. For a given phrase type, the algorithm's score is the average of 54 rho values (18 participants per group $\times$ 3 groups = 54 rho values). For 3 phrase types with 54 rho values each and 24 ratings per rho value, we have 3,888 ratings.

4.3.2 Comparison with Other Approaches

Table 20 compares the dual-space model to vector addition and element-wise multiplication. We use the development set to tune the parameters for all three approaches. For vector addition, ab is represented by $\mathbf{a} + \mathbf{b}$ and cd is represented by $\mathbf{c} + \mathbf{d}$. The similarity of ab and cd is given by the cosine of the two vectors. Element-wise multiplication uses Equation 21 to represent ab and cd. The dual-space model uses Equation 32.

The average correlation of the dual-space model (0.48) is significantly below the average correlation of vector addition using function space (0.51). Element-wise multiplication with mono space (0.47) is also significantly below vector addition using function space (0.51). The

20. The information in this paragraph is based on personal communication with Jeff Mitchell in June, 2010. Mitchell and Lapata's (2010) paper does not describe how Spearman's rho is applied.

	Correlation for each phrase type				
Algorithm	ad-nn	nn-nn	vb-ob	avg	Comment
human	0.56	0.54	0.57	0.56	leave-one-out correlation between subjects
dual-space	0.48	0.54	0.43	0.48	domain and function space
addition	0.47	0.61	0.42	0.50	mono space
addition	0.32	0.55	0.41	0.42	domain space
addition	0.49	0.55	0.48	0.51	function space
multiplication	0.43	0.57	0.41	0.47	mono space
multiplication	0.35	0.58	0.39	0.44	domain space
multiplication	0.39	0.45	0.27	0.37	function space

Table 20: Performance of the models on the evaluation dataset.

difference between the dual-space model (0.48) and element-wise multiplication with mono space (0.47) is not signficant. The average correlation for an algorithm is based on 162 rho values (3 phrase types $\times$ 3 groups $\times$ 18 participants = 162 rho values = 162 participants). We calculate the statistical significance using a paired t-test with a 95% significance level, based on 162 pairs of rho values.

4.3.3 Order Sensitivity

Mitchell and Lapata's (2010) dataset does not test for order sensitivity. Given a phrase pair $ab \sim cd$, we can test for order sensitivity by adding a new pair $ab \sim dc$. We assume that all such new pairs would be given a rating of 1 by the human participants. In Table 21, we show what happens when this transformation is applied to the examples in Table 19. To save space, we only give the examples for participant number 114.

Participant	Phrase type	Group	Phrase pair	Similarity
114	adjective-noun	2	certain circumstance ∼ particular case	6
114	adjective-noun	2	certain circumstance ∼ case particular	1
114	adjective-noun	2	large number ∼ great majority	4
114	adjective-noun	2	large number ∼ majority great	1
114	adjective-noun	2	further evidence ∼ low cost	2
114	adjective-noun	2	further evidence ∼ cost low	1

Table 21: Testing for order sensitivity by adding new phrase pairs.

Table 22 gives the results with the new, expanded dataset. With this more stringent dataset, the dual-space model performs significantly better than both vector addition and vector multiplication. Unlike element-wise multiplication and vector addition, the dual-space model addresses the issue of order sensitivity.

We manually inspected the new pairs that were automatically rated 1 and found that a rating of 1 was reasonable in all cases, although some cases could be disputed. For example, the original noun-noun pair *tax charge* ∼ *interest rate* generates the new pair *tax charge* ∼ *rate interest* and the original verb-object pair *produce effect* ∼ *achieve result* generates the new pair *produce effect* ∼ *result achieve*. It seems that we have a natural tendency to correct

Algorithm	Correlation for each phrase type ad-nn	nn-nn	vb-ob	avg	Comment
human	0.71	0.81	0.73	0.75	leave-one-out correlation between subjects
dual-space	0.66	0.37	0.62	0.55	domain and function space
addition	0.22	0.25	0.19	0.22	mono space
addition	0.15	0.22	0.18	0.18	domain space
addition	0.23	0.23	0.19	0.22	function space
multiplication	0.20	0.24	0.18	0.21	mono space
multiplication	0.18	0.22	0.18	0.19	domain space
multiplication	0.18	0.19	0.12	0.17	function space

Table 22: Performance when the dataset is expanded to test for order sensitivity.

these incorrectly ordered pairs in our minds and then assign them higher ratings than they deserve. We predict that human ratings of these new pairs would vary greatly, depending on the instructions that were given to the human raters. If the instructions emphasized the importance of word order, the new pairs would get low ratings. This prediction is supported by the results of SemEval 2012 Task 2 (Jurgens, Mohammad, Turney, & Holyoak, 2012), where the instructions to the raters emphasized the importance of word order and wrongly ordered pairs received low ratings.

4.3.4 Summary

When the dataset does not test for order sensitivity, vector addition performs slightly better than the dual-space model. When the dataset tests for order sensitivity, the dual-space model surpasses both vector addition and element-wise multiplication by a large margin.

4.4 Domain versus Function as Associated versus Similar

Chiarello et al. (1990) created a dataset of 144 word pairs that they labeled *similar-only*, *associated-only*, or *similar+associated* (48 pairs in each of the three classes). Table 23 shows some examples from their dataset. These labeled pairs were created for cognitive psychology experiments with human subjects. In their experiments, they found evidence that processing *associated* words engages the left and right hemispheres of the brain in ways that are different from processing *similar* words. That is, it seems that there is a fundamental neurological difference between these two types of semantic relatedness.[21]

We hypothesize that similarity in domain space, $\mathrm{sim}_\mathrm{d}(a, b)$, is a measure of the degree to which two words are *associated* and similarity in function space, $\mathrm{sim}_\mathrm{f}(a, b)$, is a measure of the degree to which two words are *similar*. To test this hypothesis, we define similar-only, $\mathrm{sim}_\mathrm{so}(a, b)$, associated-only, $\mathrm{sim}_\mathrm{ao}(a, b)$, and similar+associated, $\mathrm{sim}_\mathrm{sa}(a, b)$, as follows:

$$\mathrm{ratio}(x, y) = \begin{cases} x/y & \text{if } x > 0 \text{ and } y > 0 \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

21. There is some controversy among cognitive scientists over the distinction between semantic similarity and association (McRae, Khalkhali, & Hare, 2011).

Word pair	Class label
table:bed	similar-only
music:art	similar-only
hair:fur	similar-only
house:cabin	similar-only
cradle:baby	associated-only
mug:beer	associated-only
camel:hump	associated-only
cheese:mouse	associated-only
ale:beer	similar+associated
uncle:aunt	similar+associated
pepper:salt	similar+associated
frown:smile	similar+associated

Table 23: Examples of word pairs from Chiarello et al. (1990), labeled *similar-only*, *associated-only*, or *similar+associated*. The full dataset is in their Appendix.

$$\mathrm{sim_{so}}(a,b) = \mathrm{ratio}(\mathrm{sim_f}(a,b), \mathrm{sim_d}(a,b)) \tag{34}$$
$$\mathrm{sim_{ao}}(a,b) = \mathrm{ratio}(\mathrm{sim_d}(a,b), \mathrm{sim_f}(a,b)) \tag{35}$$
$$\mathrm{sim_{sa}}(a,b) = \mathrm{geo}(\mathrm{sim_d}(a,b), \mathrm{sim_f}(a,b)) \tag{36}$$

The intention is that $\mathrm{sim_{so}}$ is high when $\mathrm{sim_f}$ is high and $\mathrm{sim_d}$ is low, $\mathrm{sim_{ao}}$ is high when $\mathrm{sim_d}$ is high and $\mathrm{sim_f}$ is low, and $\mathrm{sim_{sa}}$ is high when both $\mathrm{sim_d}$ and $\mathrm{sim_f}$ are high. This is illustrated in Figure 4.

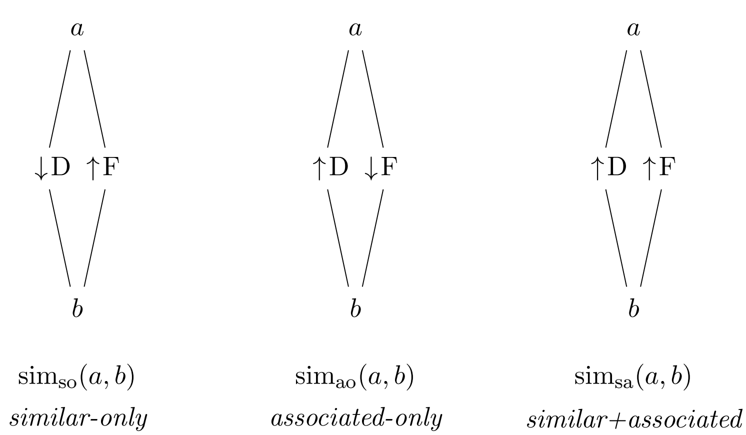


Figure 4: Diagrams of Equations 34, 35, and 36.

4.4.1 Evaluation

From the experiments in the three preceding subsections, we have three sets of parameter settings for the dual-space model. Table 24 shows these parameter values. In effect, these three sets of parameter setttings give us three variations of the similarity measures, $\mathrm{sim}_{\mathrm{so}}$, $\mathrm{sim}_{\mathrm{ao}}$, and $\mathrm{sim}_{\mathrm{sa}}$. We will evaluate the three variations to see how well they correspond to the labels in Chiarello et al.'s (1990) dataset.

Similarity	Description	Section	k_{d}	p_{d}	k_{f}	p_{f}
$\mathrm{sim}_{\mathrm{r}}(a:b,c:d)$	similarity of relations	4.1	800	-0.1	300	0.5
$\mathrm{sim}_{\mathrm{c}}(ab,c)$	similarity of noun-modifier compositions	4.2	800	0.3	100	0.6
$\mathrm{sim}_{\mathrm{p}}(ab,cd)$	similarity of phrases	4.3	200	0.3	600	0.6

Table 24: Parameter settings for the dual-space model.

For a given similarity measure, such as $\mathrm{sim}_{\mathrm{so}}$, we can sort the 144 word pairs in descending order of their similarities and then look at the top N pairs to see how many of them have the desired label; in the case of $\mathrm{sim}_{\mathrm{so}}$, we would like to see that the majority of the top N have the label *similar-only*. Table 25 shows the percentage of pairs that have the desired labels for each of the three variations of the three similarity measures. Note that random guessing would yield 33%, since the three classes of pairs have the same size.

		Percentage of top N with desired label		
Source of parameters	N	similar-only	associated-only	similar+associated
$\mathrm{sim}_{\mathrm{r}}(a:b,c:d)$	10	70	90	90
	20	80	85	80
	30	63	77	73
$\mathrm{sim}_{\mathrm{c}}(ab,c)$	10	90	90	80
	20	80	70	70
	30	70	67	73
$\mathrm{sim}_{\mathrm{p}}(ab,cd)$	10	50	90	80
	20	65	80	80
	30	47	77	73

Table 25: Percentage of the top N word pairs with the desired labels.

For all three sets of parameter settings, Table 25 displays a high density of the desired labels at the tops of the sorted lists. The density slowly decreases as we move down the lists. This is evidence that the three similarity measures are capturing the three classes of Chiarello et al. (1990).

As another test of the hypothesis, we use the three similarity measures to create feature vectors of three elements for each word pair. That is, the word pair $a:b$ is represented by the feature vector $\langle \mathrm{sim}_{\mathrm{so}}(a,b), \mathrm{sim}_{\mathrm{ao}}(a,b), \mathrm{sim}_{\mathrm{sa}}(a,b)\rangle$. We then use supervised learning with ten-fold cross-validation to classify the feature vectors into the three classes of Chiarello et al. (1990). For the learning algorithm, we use logistic regression, as implemented in

Weka.[22] The results are summarized in Table 26. These results lend further support to the hypothesis that similarity in domain space, $\mathrm{sim_d}(a, b)$, is a measure of the degree to which two words are *associated* and similarity in function space, $\mathrm{sim_f}(a, b)$, is a measure of the degree to which two words are *similar*.

		F-measure			
Source of parameters	Accuracy	similar-only	associated-only	similar+associated	average
$\mathrm{sim_r}(a:b, c:d)$	61.1	0.547	0.660	0.625	0.611
$\mathrm{sim_c}(ab, c)$	59.0	0.583	0.702	0.490	0.592
$\mathrm{sim_p}(ab, cd)$	58.3	0.472	0.699	0.563	0.578

Table 26: Performance of logistic regression with the three similarity measures as features.

In Table 25, similar-only seems more sensitive to the parameter settings than associated-only and similar+associated. We hypothesize that this is because function similarity is more difficult to measure than domain similarity. Note that the construction of function space (Section 3.3) is more complex than the construction of domain space (Section 3.2). Intuitively, it seems easier to identify the domain of a thing than to identify its functional role. Gentner's (1991) work suggests that children master domain similarity before they become competent with function similarity.

5. Discussion of Experiments

This section discusses the results of the previous section.

5.1 Summary of Results

In Section 4.1, we used 374 multiple-choice analogy questions to evaluate the dual-space model of relational similarity, $\mathrm{sim_r}(a : b, c : d)$. The difference between the performance of the dual-space model (51.1% accuracy) and the best past result (56.1% accuracy), using a holistic model, was not statistically significant. Experiments with a reformulated version of the questions, designed to test order sensitivity, supported the hypothesis that both domain and function space are required. Function space by itself is not sensitive to order and merging the two spaces (mono space) causes a significant drop in performance.

In Section 4.2, we automatically generated 2,180 multiple-choice noun-modifier composition questions with WordNet, to evaluate the dual-space model of noun-modifier compositional similarity, $\mathrm{sim_c}(ab, c)$. The difference between the performance of the dual-space model (58.3% accuracy) and the state-of-the-art element-wise multiplication model (57.5% accuracy) was not statistically significant. The best performance was obtained with a holistic model (81.6%), but this model does not address the issue of linguistic creativity. Further experiments suggest that a significant fraction of the gap between the holistic model and the other models is due to noncompositional phrases. A limitation of the element-wise multiplication model is lack of sensitivity to order. Experiments with a reformulated version

22. Weka is available at http://www.cs.waikato.ac.nz/ml/weka/.

of the questions, designed to test order sensitivitiy, demonstrated a statistically significant advantage to the dual-space model over the element-wise multiplication and vector addition models.

In Section 4.3, we used Mitchell and Lapata's (2010) dataset of 324 pairs of phrases to evaluate the dual-space model of phrasal similarity, $\mathrm{sim_p}(ab, cd)$. A reformulated version of the dataset, modified to test order sensitivitiy, showed a statistically significant advantage to the dual-space model over the element-wise multiplication and vector addition models.

In Section 4.4, we used Chiarello et al.'s (1990) dataset of 144 word pairs, labeled *similar-only*, *associated-only*, or *similar+associated*, to test the hypothesis that similarity in domain space, $\mathrm{sim_d}(a, b)$, is a measure of the degree to which two words are *associated* and similarity in function space, $\mathrm{sim_f}(a, b)$, is a measure of the degree to which two words are *similar*. The experimental results support the hypothesis. This is interesting because Chiarello et al. (1990) argue that there is a fundamental neurological difference in the way people process these two kinds of semantic relatedness.

The experiments support the claim that the dual-space model can address the issues of linguistic creativity, order sensitivity, and adaptive capacity. Furthermore, the dual-space model provides a unified approach to both semantic relations and semantic composition.

5.2 Corpus-based Similarity versus Lexicon-based Similarity

The results in Section 4.4 suggest that function similarity may correspond to the kind of taxonomical similarity that is often associated with lexicons, such as WordNet (Resnik, 1995; Jiang & Conrath, 1997; Leacock & Chodrow, 1998; Hirst & St-Onge, 1998). The word pairs in Table 23 that are labeled *similar-only* are the kinds of words that typically share a common hypernym in a taxonomy. For example, *table:bed* share the hypernym *furniture*. We believe that this is correct, but it does not necessarily imply that lexicon-based similarity measures would be better than a corpus-based approach, such as we have used here.

Of the various similarities in Section 4, arguably relational similarity, $\mathrm{sim_r}(a\!:\!b, c\!:\!d)$, makes the most use of function similarity. By itself, function similarity achieves 50.8% on the SAT questions (original five-choice version; see Table 12). However, the best performance achieved on the SAT questions using WordNet is 43.0% (Veale, 2004). The difference is statistically significant at the 95% confidence level, based on Fisher's Exact Test.

Consider the analogy *traffic* is to *street* as *water* is to *riverbed*. One of the SAT questions involves this analogy, with *traffic*:*street* as the stem pair and *water*:*riverbed* as the correct choice. Both $\mathrm{sim_r}(a : b, c : d)$ (Equation 25) and function similarity by itself (Equation 22) make the correct choice. We can recognize that *traffic* and *water* have a high degree of function similarity; in fact, this similarity is used in hydrodynamic models of traffic flow (Daganzo, 1994). However, we must climb the WordNet hierachy all the way up to *entity* before we find a shared hypernym for *traffic* and *water*. We believe that no manually generated lexicon can capture all of the functional similarity that can be discovered in a large corpus.

6. Theoretical Considerations

This section examines some theoretical questions about the dual-space model.

6.1 Vector Composition versus Similarity Composition

In the dual-space model, a phrase has no stand-alone, general-purpose representation, as a composite phrase, apart from the representations of the component words. The composite meaning is constructed in the context of a given task. For example, if the task is to measure the similarity of the relation in *dog*:*house* to the relation in *bird*:*nest*, then we compose the meanings of *dog* and *house* one way (see Section 4.1); if the task is to measure the similarity of the phrase *dog house* to the word *kennel*, then we compose the meanings of *dog* and *house* another way (see Section 4.2); if the task is to measure the similarity of the phrase *dog house* to the phrase *canine shelter*, then we compose the meanings of *dog* and *house* a third way (see Section 4.3). The composition is a construction that explicitly ties together the two things that are being compared, and it depends on the nature of the comparison that is desired, the task that is to be performed. We hypothesize that no single stand-alone, task-independent representation can be constructed that is suitable for all purposes.

As we noted in the introduction, composition of vectors can result in a stand-alone representation of a phrase, but composing similarities necessarily yields a linking structure that connects a phrase to other phrases. These linking structures can be seen in Figures 1 to 4. Intuitively, it seems that an important part of how we understand a phrase is by connecting it to other phrases. Part of our understanding of *dog house* is its connection to *kennel*. Dictionaries make these kinds of connections explicit. From this perspective, the idea of an explicit linking structure seems natural, given that making connnections among words and phrases is an essential aspect of meaning and understanding.

6.2 General Form of Similarities in the Dual-Space Model

In this subsection, we present a general scheme that ties together the various similarities that were defined in Section 4. This scheme includes similarities between chunks of text of arbitrary size. The scheme encompasses phrasal similarity, relational similarity, and compositional similarity.

Let $\mathbf{t}$ be a chunk of text (an ordered set of words), $\langle t_1, t_2, \ldots, t_n \rangle$, where each t_i is a word. We represent the semantics of $\mathbf{t}$ by $T = \langle \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{D}$ and $\mathbf{F}$ are matrices. Each row vector $\mathbf{d}_i$ in $\mathbf{D}$, $i = 1, 2, \ldots, n$, is the row vector in domain space that represents the domain semantics of the word t_i. Each row vector $\mathbf{f}_i$ in $\mathbf{F}$, $i = 1, 2, \ldots, n$, is the row vector in function space that represents the function semantics of the word t_i. To keep the notation simple, the parameters, k_d and p_d for domain space and k_f and p_f for function space, are implicit. Assume that the row vectors in $\mathbf{D}$ and $\mathbf{F}$ are normalized to unit length. Note that the size of the representation T scales linearly with n, the number of words in $\mathbf{t}$, hence we have *information scalability*. For large values of n, there will inevitably be duplicate words in $\mathbf{t}$, so the representation could easily be compressed to sublinear size without loss of information.

Let $\mathbf{t}_1$ and $\mathbf{t}_2$ be two chunks of text with representations $T_1 = \langle \mathbf{D}_1, \mathbf{F}_1 \rangle$ and $T_2 = \langle \mathbf{D}_2, \mathbf{F}_2 \rangle$, where $\mathbf{t}_1$ contains n_1 words and $\mathbf{t}_2$ has n_2 words. Let $\mathbf{D}_1$ and $\mathbf{D}_2$ have the same parameters, k_d and p_d, and let $\mathbf{F}_1$ and $\mathbf{F}_2$ have the same parameters, k_f and p_f. Then $\mathbf{D}_1$ is $n_1 \times k_\mathrm{d}$, $\mathbf{D}_2$ is $n_2 \times k_\mathrm{d}$, $\mathbf{F}_1$ is $n_1 \times k_\mathrm{f}$, and $\mathbf{F}_2$ is $n_2 \times k_\mathrm{f}$. Note that $\mathbf{D}_1\mathbf{D}_1^\mathsf{T}$ is an $n_1 \times n_1$ matrix of the cosines between any two row vectors in $\mathbf{D}_1$. That is, the element in the i-th row and j-th column of $\mathbf{D}_1\mathbf{D}_1^\mathsf{T}$ is $\cos(\mathbf{d}_i, \mathbf{d}_j)$. Likewise, $\mathbf{D}_1\mathbf{D}_2^\mathsf{T}$ is an $n_1 \times n_2$ matrix of the cosines between any row vector in $\mathbf{D}_1$ and any row vector in $\mathbf{D}_2$.

Suppose that we wish to measure the similarity, $\mathrm{sim}(\mathbf{t}_1, \mathbf{t}_2)$, between the two chunks of text, $\mathbf{t}_1$ and $\mathbf{t}_2$. In this paper, we have restricted the similarity measures to the following general form:

$$\mathrm{sim}(\mathbf{t}_1, \mathbf{t}_2) = f(\mathbf{D}_1\mathbf{D}_1^\mathsf{T}, \mathbf{D}_1\mathbf{D}_2^\mathsf{T}, \mathbf{D}_2\mathbf{D}_2^\mathsf{T}, \mathbf{F}_1\mathbf{F}_1^\mathsf{T}, \mathbf{F}_1\mathbf{F}_2^\mathsf{T}, \mathbf{F}_2\mathbf{F}_2^\mathsf{T}) \tag{37}$$

In other words, the only input to the composition function f is cosines (and the implicit parameters, k_d, p_d, k_f, and p_f); f does not operate directly on any of the row vectors in $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{F}_1$, and $\mathbf{F}_2$. In contrast to much of the work discussed in Section 2.1, the composition operation is shifted out of the representations, T_1 and T_2, and into the similarity measure, f. The exact specification of f depends on the task at hand. When T_1 and T_2 are sentences, we envision that the structure of f will be determined by the syntactic structures of the two sentences.[23]

Consider relational similarity (Section 4.1):

$$\mathrm{sim}_1(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim}_\mathrm{f}(a,c), \mathrm{sim}_\mathrm{f}(b,d)) \tag{38}$$

$$\mathrm{sim}_2(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim}_\mathrm{d}(a,b), \mathrm{sim}_\mathrm{d}(c,d)) \tag{39}$$

$$\mathrm{sim}_3(a\!:\!b, c\!:\!d) = \mathrm{geo}(\mathrm{sim}_\mathrm{d}(a,d), \mathrm{sim}_\mathrm{d}(c,b)) \tag{40}$$

$$\mathrm{sim}_\mathrm{r}(a\!:\!b, c\!:\!d) = \begin{cases} \mathrm{sim}_1(a\!:\!b, c\!:\!d) & \text{if } \mathrm{sim}_2(a\!:\!b, c\!:\!d) \geq \mathrm{sim}_3(a\!:\!b, c\!:\!d) \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

This fits the form of Equation 37 when we have $\mathbf{t}_1 = \langle a, b \rangle$ and $\mathbf{t}_2 = \langle c, d \rangle$. We can see that sim_1 is based on cosines from $\mathbf{F}_1\mathbf{F}_2^\mathsf{T}$, sim_2 is based on cosines from $\mathbf{D}_1\mathbf{D}_1^\mathsf{T}$ and $\mathbf{D}_2\mathbf{D}_2^\mathsf{T}$, and sim_3 is based on cosines from $\mathbf{D}_1\mathbf{D}_2^\mathsf{T}$.

Consider compositional similarity (Section 4.2):

$$\mathrm{sim}_1(ab, c) = \mathrm{geo}(\mathrm{sim}_\mathrm{d}(a,c), \mathrm{sim}_\mathrm{d}(b,c), \mathrm{sim}_\mathrm{f}(b,c)) \tag{42}$$

$$\mathrm{sim}_\mathrm{c}(ab, c) = \begin{cases} \mathrm{sim}_1(ab, c) & \text{if } a \neq c \text{ and } b \neq c \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

This can be seen as an instance of Equation 37 in which $\mathbf{t}_1 = \langle a, b \rangle$ and $\mathbf{t}_2 = \langle c \rangle$. In this case, sim_1 is based on cosines from $\mathbf{D}_1\mathbf{D}_2^\mathsf{T}$ and $\mathbf{F}_1\mathbf{F}_2^\mathsf{T}$. The constraints, $a \neq c$ and $b \neq c$, can be expressed in terms of cosines from $\mathbf{D}_1\mathbf{D}_2^\mathsf{T}$, as $\mathrm{sim}_\mathrm{d}(a,c) \neq 1$ and $\mathrm{sim}_\mathrm{d}(b,c) \neq 1$. (Equivalently, we could use cosines from $\mathbf{F}_1\mathbf{F}_2^\mathsf{T}$.) Similar analyses apply to the similarities in Sections 4.3 and 4.4; these similarities are also instances of Equation 37.

Although the representations T_1 and T_2 have sizes that are linear functions of the numbers of phrases in $\mathbf{t}_1$ and $\mathbf{t}_2$, the size of the composition in Equation 37 is a quadratic function of the numbers of phrases in $\mathbf{t}_1$ and $\mathbf{t}_2$. However, specific instances of this general equation may be less than quadratic in size, and it may be possible to limit the growth

23. Note that there is no requirement for the two chunks of text, $\mathbf{t}_1$ and $\mathbf{t}_2$, to have the same number of words. That is, n_1 does not necessarily equal n_2. In Section 4.2, $n_1 \neq n_2$.

to a linear function. Also, in general, quadratic growth is often acceptable in practical applications (Garey & Johnson, 1979).

With function words (e.g., prepositions, conjunctions), one option would be to treat them the same as any other words. They would be represented by vectors and their similarities would be calculated in function and domain spaces. Another possibility would be to use function words as hints to guide the construction of the composition function f. The function words would not correspond to vectors; instead they would contribute to determining the linking structure that connects the two given chunks of text. The first option appears more elegant, but the choice between the options should be made empirically.

6.3 Automatic Composition of Similarities

In Section 4, we manually constructed the functions that combined the similarity measures, using our intuition and background knowledge. Manual construction will not scale up to the task of comparing any two arbitrarily chosen sentences. However, there are good reasons for believing that the construction of composition functions can be automated.

Turney (2008a) presents an algorithm for solving analogical mapping problems, such as the analogy between the solar system and the Rutherford-Bohr model of the atom. Given a list of terms from the solar system domain, {*planet, attracts, revolves, sun, gravity, solar system, mass*}, and a list of terms from the atomic domain, {*revolves, atom, attracts, electromagnetism, nucleus, charge, electron*}, it can automatically generate a one-to-one mapping from one domain to the other, {*solar system* → *atom, sun* → *nucleus, planet* → *electron, mass* → *charge, attracts* → *attracts, revolves* → *revolves, gravity* → *electromagnetism*}. On twenty analogical mapping problems, it attains an accuracy of 91.5%, compared to an average human accuracy of 87.6%.

The algorithm scores the quality of a candidate analogical mapping by composing the similarities of the mapped terms. The composition function is addition and the individual component similarities are holistic relational similarities. The algorithm searches through the space of possible mappings for the mapping that maximizes the composite similarity measure. That is, analogical mapping is treated as an *argmax* problem, where the argument to be maximized is a mapping function. In effect, the output of the algorithm (an analogical mapping) is an automically generated composition of similarities. The mapping structures found by the algorithm are essentially the same as the linking structures that we see in Figures 1 to 4.

We believe that a variation of Turney's (2008a) algorithm could be used to automatically compose similarities in the dual-space model; for example, it should be possible to identify paraphrases using automatic similarity composition. The proposal is to search for a composition that maximizes composite similarity, subject to various constraints (such as constraints based on the syntax of the sentences). Turney (2008a) points out that analogical mapping could be used to align the words in two sentences, but does not experimentally evaluate this suggestion.

Recent work (Lin & Bilmes, 2011) has shown that *argmax* problems can be solved efficiently and effectively if they can be framed as monotone submodular function maximization problems. We believe that automatic composition of similarities can fit naturally into this framework, which would result in highly scalable algorithms for semantic composition.

Regarding information scalability, the dual-space model does not suffer from information loss (unlike approaches that represent compositions with vectors of fixed dimensionality), because the sizes of the representations grow as the lengths of the phrases grow. The growth might be quadratic, but it is not exponential. There are questions about how to automate composition of similarities, which may have an impact on the computational complexity of scaling to longer phrases, but there is evidence that these questions are tractable.

7. Limitations and Future Work

One area for future work is to experiment with longer phrases (more than two words) and sentences, as discussed in Section 6.3. An interesting topic for research is how parsing might be used to constrain the automatic search for similarity composition functions.

Here we have focused on two spaces, domain and function, but it seems likely to us that a model with more spaces would yield better performance. We are currently experimenting with a quad-space model that includes domain (noun-based contextual patterns), function (verb-based), quality (adjective-based), and manner (adverb-based) spaces. The preliminary results with quad-space are promising. Quad-space seems to be related to Pustejovsky's (1991) four-part qualia structure.

Another issue we have avoided here is morphology. As discussed in Section 3.6, we used the *validForms* function in the WordNet::QueryData Perl interface to WordNet to map morphological variations of words to their base forms. This implies that, for example, a singular noun and its plural form should have the same semantic representation. This is certainly a simplification and a more sophisticated model would use different representations for different morphological forms of a word.

We have also avoided the issue of polysemy. It should be possible to extend past work with polysemy in VSMs to the dual-space model (Schütze, 1998; Pantel & Lin, 2002; Erk & Padó, 2008).

In this paper, we have treated the holistic model and the dual-space model as if they are competitors, but there are certain cases, such as idiomatic expressions, where the holistic approach is required. Likewise, the holistic approach is limited by its inability to handle linguistic creativity. These considerations suggest that the holistic and dual-space models must be integrated. This is another topic for future work.

Arguably it is a limitation of the dual-space model that there are four parameters to tune (k_{d}, p_{d}, k_{f}, and p_{f}). On the other hand, perhaps any model with adaptive capacity must have some parameters to tune. Further research is needed.

A number of design decisions were made in the construction of domain and function space, especially in the conversion of phrases to contextual patterns (Sections 3.2 and 3.3). These decisions were guided by our intuitions. We expect that the exploration and experimental evaluation of this design space will be a fruitful area for future research.

The construction of function space (Section 3.3) is specific to English. It may generalize readily to other Indo-European languages, but some other languages may present a challenge. This is another topic for future research.

Most of our composite similarities use the geometric mean to combine domain and function similarities, but we see no reason to restrict the possible composition functions.

Equation 37 allows any composition function f. Exploring the space of possible composition functions is another topic for future work.

Another question is how formal logic and textual entailment can be integrated into this approach. The dual-space model seems to be suitable for recognizing paraphrases, but there is no obvious way to handle entailment. More generally, we have focused on various kinds of similarity, but when we scale up from phrases (*red ball*) to sentences (*The ball is red*), we encounter truth and falsity. Gärdenfors (2004) argues that spatial models are a bridge between low-level connectionist models and high-level symbolic models. He claims that spatial models are best for questions about similarity and symbolic models are best for questions about truth. We do not yet know how to join these two kinds of models.

8. Conclusions

The goal in this research has been to develop a model that unifies semantic relations and compositions, while also addressing linguistic creativity, order sensitivity, adaptive capacity, and information scalability. We believe that the dual-space model achieves this goal, although there is certainly room for improvement and further research.

There are many kinds of word–context matrices, based on various notions of context; Sahlgren (2006) gives a good overview of the types of context that have been explored in past work. The novelty of the dual-space model is that it includes two distinct and complementary word–context matrices that work together synergistically.

With two distinct spaces, we have two distinct similarity measures, which can be combined in many different ways. With multiple similarity measures, similarity composition becomes a viable alternative to vector composition. For example, instead of multiplying vectors, such as $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$, we can multiply similarities, such as $\mathrm{sim}_{\mathrm{sa}}(a, b) = \mathrm{geo}(\mathrm{sim}_{\mathrm{d}}(a, b), \mathrm{sim}_{\mathrm{f}}(a, b))$. The results here suggest that this is a fruitful new way to look at some of the problems of semantics.

Acknowledgments

Thanks to George Foster, Yair Neuman, David Jurgens, and the reviewers of *JAIR* for their very helpful comments on an earlier version of this paper. Thanks to Charles Clarke for the corpus used to build the three spaces, to Stefan Büttcher for Wumpus, to the creators of WordNet for making their lexicon available, to the developers of OpenNLP, to Doug Rohde for SVDLIBC, to Jeff Mitchell and Mirella Lapata for sharing their data and answering questions about their evaluation methodology, to Christine Chiarello, Curt Burgess, Lorie Richards, and Alma Pollock for making their data available, to Jason Rennie for the WordNet::QueryData Perl interface to WordNet, and to the developers of Perl Data Language.

References

Aerts, D., & Czachor, M. (2004). Quantum aspects of semantic analysis and symbolic artificial intelligence. *Journal of Physics A: Mathematical and General*, *37*, L123–L132.

Baroni, M., & Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pp. 1183–1193.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Biçici, E., & Yuret, D. (2006). Clustering word pairs to answer analogy questions. In *Proceedings of the Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006)*, Akyaka, Mugla, Turkey.

Biemann, C., & Giesbrecht, E. (2011). Distributional semantics and compositionality 2011: Shared task description and results. In *Proceedings of the Workshop on Distributional Semantics and Compositionality (DiSCo 2011)*, pp. 21–28, Portland, Oregon.

Bollegala, D., Matsuo, Y., & Ishizuka, M. (2009). Measuring the similarity between implicit semantic relations from the Web. In *Proceedings of the 18th International Conference on World Wide Web (WWW 2009)*, pp. 651–660.

Brants, T., & Franz, A. (2006). *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

Bullinaria, J., & Levy, J. (2007). Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, *39*(3), 510–526.

Büttcher, S., & Clarke, C. (2005). Efficiency vs. effectiveness in terabyte-scale information retrieval. In *Proceedings of the 14th Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD.

Caron, J. (2001). Experiments with LSA scoring: Optimal rank and basis.. In *Proceedings of the SIAM Computational Information Retrieval Workshop*, pp. 157–169, Raleigh, NC.

Chiarello, C., Burgess, C., Richards, L., & Pollock, A. (1990). Semantic and associative priming in the cerebral hemispheres: Some words do, some words don't ... sometimes, some places. *Brain and Language*, *38*, 75–104.

Chomsky, N. (1975). *The Logical Structure of Linguistic Theory*. Plenum Press.

Church, K., & Hanks, P. (1989). Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pp. 76–83, Vancouver, British Columbia.

Clark, S., Coecke, B., & Sadrzadeh, M. (2008). A compositional distributional model of meaning. In *Proceedings of the 2nd Symposium on Quantum Interaction*, pp. 133–140, Oxford, UK.

Clark, S., & Pulman, S. (2007). Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pp. 52–55, Stanford, CA.

Conway, J. H., & Sloane, N. J. A. (1998). *Sphere Packings, Lattices and Groups*. Springer.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, *28*(4), 269–287.

Davidov, D., & Rappoport, A. (2008). Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proceedings of the 46th Annual Meeting of the ACL and HLT (ACL-HLT-08)*, pp. 692–700, Columbus, Ohio.

Erk, K., & Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pp. 897–906, Honolulu, HI.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database*. MIT Press.

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In *Studies in Linguistic Analysis*, pp. 1–32. Blackwell, Oxford.

Fodor, J., & Lepore, E. (2002). *The Compositionality Papers*. Oxford University Press.

Gärdenfors, P. (2004). *Conceptual Spaces: The Geometry of Thought*. MIT Press.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*(2), 155–170.

Gentner, D. (1991). Language and the career of similarity. In Gelman, S., & Byrnes, J. (Eds.), *Perspectives on Thought and Language: Interrelations in Development*, pp. 225–277. Cambridge University Press.

Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations* (Third edition). Johns Hopkins University Press, Baltimore, MD.

Grefenstette, E., & Sadrzadeh, M. (2011). Experimenting with transitive verbs in a DisCoCat. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*.

Grice, H. P. (1989). *Studies in the Way of Words*. Harvard University Press, Cambridge, MA.

Guevara, E. (2010). A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics (GEMS 2010)*, pp. 33–37.

Harris, Z. (1954). Distributional structure. *Word*, *10*(23), 146–162.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics (COLING-92)*, pp. 539–545.

Herdağdelen, A., & Baroni, M. (2009). Bagpack: A general framework to represent semantic relations. In *Proceedings of the EACL 2009 Geometrical Models for Natural Language Semantics (GEMS) Workshop*, pp. 33–40.

Hirst, G., & St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. In Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*, pp. 305–332. MIT Press.

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, pp. 19–33, Tapei, Taiwan.

Johannsen, A., Alonso, H. M., Rishøj, C., & Søgaard, A. (2011). Shared task system description: Frustratingly hard compositionality prediction. In *Proceedings of the Workshop on Distributional Semantics and Compositionality (DiSCo 2011)*, pp. 29–32, Portland, Oregon.

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological review*, *114*, 1–37.

Jurgens, D. A., Mohammad, S. M., Turney, P. D., & Holyoak, K. J. (2012). SemEval-2012 Task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pp. 356–364, Montréal, Canada.

Kintsch, W. (2000). Metaphor comprehension: A computational theory. *Psychonomic Bulletin & Review*, *7*(2), 257–266.

Kintsch, W. (2001). Predication. *Cognitive Science*, *25*(2), 173–202.

Kolda, T., & Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, *51*(3), 455–500.

Landauer, T. K. (2002). On the computational basis of learning and cognition: Arguments from LSA. In Ross, B. H. (Ed.), *The Psychology of Learning and Motivation: Advances in Research and Theory*, Vol. 41, pp. 43–84. Academic Press.

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

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2007). *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum, Mahwah, NJ.

Leacock, C., & Chodrow, M. (1998). Combining local context and WordNet similarity for word sense identification. In Fellbaum, C. (Ed.), *WordNet: An Electronic Lexical Database*. MIT Press.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by nonnegative matrix factorization. *Nature*, *401*, 788–791.

Lepage, Y., & Shin-ichi, A. (1996). Saussurian analogy: A theoretical account and its application. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*, pp. 717–722.

Lin, H., & Bilmes, J. (2011). A class of submodular functions for document summarization. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pp. 510–520.

Mangalath, P., Quesada, J., & Kintsch, W. (2004). Analogy-making as predication using relational information and LSA vectors. In *Proceedings of the 26th Annual Meeting of the Cognitive Science Society*, p. 1623, Austin, TX.

McRae, K., Khalkhali, S., & Hare, M. (2011). Semantic and associative relations in adolescents and young adults: Examining a tenuous dichotomy. In Reyna, V., Chapman, S., Dougherty, M., & Confrey, J. (Eds.), *The Adolescent Brain: Learning, Reasoning, and Decision Making*, pp. 39–66. APA, Washington, DC.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pp. 236–244, Columbus, Ohio. Association for Computational Linguistics.

Mitchell, J., & Lapata, M. (2010). Composition in distributional models of semantics. *Cognitive Science*, *34*(8), 1388–1429.

Moschitti, A., & Quarteroni, S. (2008). Kernels on linguistic structures for answer extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, p. 113116, Columbus, OH.

Nakov, P., & Hearst, M. (2006). Using verbs to characterize noun-noun relations. In *Proceedings of the 12th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2006)*, pp. 233–244, Varna, Bulgaria.

Nakov, P., & Hearst, M. (2007). UCB: System description for SemEval Task 4. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval 2007)*, pp. 366–369, Prague, Czech Republic.

Nastase, V., Sayyad-Shirabad, J., Sokolova, M., & Szpakowicz, S. (2006). Learning noun-modifier semantic relations with corpus-based and WordNet-based features. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pp. 781–786.

Nastase, V., & Szpakowicz, S. (2003). Exploring noun-modifier semantic relations. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, pp. 285–301, Tilburg, The Netherlands.

Niwa, Y., & Nitta, Y. (1994). Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th International Conference On Computational Linguistics*, pp. 304–309, Kyoto, Japan.

Ó Séaghdha, D., & Copestake, A. (2009). Using lexical and relational similarity to classify semantic relations. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, Athens, Greece.

Pantel, P., & Lin, D. (2002). Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 613–619, Edmonton, Canada.

Plate, T. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, *17*(4), 409–441.

Rapp, R. (2003). Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pp. 315–322.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 448–453, San Mateo, CA. Morgan Kaufmann.

Rosario, B., & Hearst, M. (2001). Classifying the semantic relations in noun-compounds via a domain-specific lexical hierarchy. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-01)*, pp. 82–90.

Rosario, B., Hearst, M., & Fillmore, C. (2002). The descent of hierarchy, and selection in relational semantics. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pp. 247–254.

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. thesis, Department of Linguistics, Stockholm University.

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project. Tech. rep., Department of Computer and Information Science, University of Pennsylvania. (3rd revision, 2nd printing).

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

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

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., & Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems (NIPS 2011)*, pp. 801–809.

Socher, R., Manning, C. D., & Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop.*

Thater, S., Fürstenau, H., & Pinkal, M. (2010). Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 948–957.

Turney, P. D. (2001). Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-01)*, pp. 491–502, Freiburg, Germany.

Turney, P. D. (2006a). Expressing implicit semantic relations without supervision. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (Coling/ACL-06)*, pp. 313–320, Sydney, Australia.

Turney, P. D. (2006b). Similarity of semantic relations. *Computational Linguistics*, *32*(3), 379–416.

Turney, P. D. (2008a). The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, *33*, 615–655.

Turney, P. D. (2008b). A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 905–912, Manchester, UK.

Turney, P. D., & Littman, M. L. (2005). Corpus-based learning of analogies and semantic relations. *Machine Learning*, *60*(1–3), 251–278.

Turney, P. D., Littman, M. L., Bigham, J., & Shnayder, V. (2003). Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pp. 482–489, Borovets, Bulgaria.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, *37*, 141–188.

Utsumi, A. (2009). Computational semantics of noun compounds in a semantic space model. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 1568–1573.

Veale, T. (2004). WordNet sits the SAT: A knowledge-based approach to lexical analogy. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pp. 606–612, Valencia, Spain.

Widdows, D. (2008). Semantic vector products: Some initial investigations. In *Proceedings of the 2nd Symposium on Quantum Interaction*, Oxford, UK.