Distributional Memory: A General Framework for Corpus-Based Semantics

Marco Baroni*
University of Trento

Alessandro Lenci**
University of Pisa

Research into corpus-based semantics has focused on the development of ad hoc models that treat single tasks, or sets of closely related tasks, as unrelated challenges to be tackled by extracting different kinds of distributional information from the corpus. As an alternative to this "one task, one model" approach, the Distributional Memory framework extracts distributional information once and for all from the corpus, in the form of a set of weighted word-link-word tuples arranged into a third-order tensor. Different matrices are then generated from the tensor, and their rows and columns constitute natural spaces to deal with different semantic problems. In this way, the same distributional information can be shared across tasks such as modeling word similarity judgments, discovering synonyms, concept categorization, predicting selectional preferences of verbs, solving analogy problems, classifying relations between word pairs, harvesting qualia structures with patterns or example pairs, predicting the typical properties of concepts, and classifying verbs into alternation classes. Extensive empirical testing in all these domains shows that a Distributional Memory implementation performs competitively against task-specific algorithms recently reported in the literature for the same tasks, and against our implementations of several state-of-the-art methods. The Distributional Memory approach is thus shown to be tenable despite the constraints imposed by its multi-purpose nature.

1. Introduction

The last two decades have seen a rising wave of interest among computational linguists and cognitive scientists in corpus-based models of semantic representation (Grefenstette 1994; Lund and Burgess 1996; Landauer and Dumais 1997; Schütze 1997; Sahlgren 2006; Bullinaria and Levy 2007; Griffiths, Steyvers, and Tenenbaum 2007; Padó and Lapata 2007; Lenci 2008; Turney and Pantel 2010). These models, variously known as vector spaces, semantic spaces, word spaces, corpus-based semantic models, or, using the term we will adopt, **distributional semantic models** (DSMs), all rely on some version of the distributional hypothesis (Harris 1954; Miller and Charles 1991), stating that the degree of semantic similarity between two words (or other linguistic units) can be modeled

* Center for Mind/Brain Sciences (CIMeC), University of Trento, C.so Bettini 31, 38068 Rovereto (TN), Italy. E-mail: marco.baroni@unitn.it.
** Department of Linguistics T. Bolelli, University of Pisa, Via Santa Maria 36, 56126 Pisa (PI), Italy. E-mail: alessandro.lenci@ling.unipi.it.

Submission received: 11 January 2010; revised submission received: 15 April 2010; accepted for publication: 1 June 2010.

as a function of the degree of overlap among their linguistic contexts. Conversely, the format of distributional representations greatly varies depending on the specific aspects of meaning they are designed to model.

The most straightforward phenomenon tackled by DSMs is what Turney (2006b) calls **attributional similarity**, which encompasses standard taxonomic semantic relations such as synonymy, co-hyponymy, and hypernymy. Words like *dog* and *puppy*, for example, are attributionally similar in the sense that their meanings share a large number of attributes: They are animals, they bark, and so on. Attributional similarity is typically addressed by DSMs based on word collocates (Grefenstette 1994; Lund and Burgess 1996; Schütze 1997; Bullinaria and Levy 2007; Padó and Lapata 2007). These collocates are seen as proxies for various attributes of the concepts that the words denote. Words that share many collocates denote concepts that share many attributes. Both *dog* and *puppy* may occur near *owner*, *leash*, and *bark*, because these words denote properties that are shared by dogs and puppies. The attributional similarity between *dog* and *puppy*, as approximated by their contextual similarity, will be very high.

DSMs succeed in tasks like synonym detection (Landauer and Dumais 1997) or concept categorization (Almuhareb and Poesio 2004) because such tasks require a measure of attributional similarity that favors concepts that share many properties, such as synonyms and co-hyponyms. However, many other tasks require detecting different kinds of semantic similarity. Turney (2006b) defines **relational similarity** as the property shared by pairs of words (e.g, *dog–animal* and *car–vehicle*) linked by similar semantic relations (e.g., hypernymy), despite the fact that the words in one pair might not be attributionally similar to those in the other pair (e.g., *dog* is not attributionally similar to *car*, nor is *animal* to *vehicle*). Turney generalizes DSMs to tackle relational similarity and represents pairs of words in the space of the patterns that connect them in the corpus. Pairs of words that are connected by similar patterns probably hold similar relations, that is, they are relationally similar. For example, we can hypothesize that *dog–tail* is more similar to *car–wheel* than to *dog–animal*, because the patterns connecting *dog* and *tail* (*of*, *have*, etc.) are more like those of *car–wheel* than like those of *dog–animal* (*is a*, *such as*, etc.). Turney uses the relational space to implement tasks such as solving analogies and harvesting instances of relations. Although they are not explicitly expressed in these terms, relation extraction algorithms (Hearst 1992, 1998; Girju, Badulescu, and Moldovan 2006; Pantel and Pennacchiotti 2006) also rely on relational similarity, and focus on learning one relation type at a time (e.g., finding parts).

Although semantic similarity, either attributional or relational, has the lion's share in DSMs, similarity is not the only aspect of meaning that is addressed by distributional approaches. For instance, the notion of property plays a key role in cognitive science and linguistics, which both typically represent concepts as clusters of properties (Jackendoff 1990; Murphy 2002). In this case, the task is not to find out that *dog* is similar to *puppy* or *cat*, but that it has a *tail*, it is used for *hunting*, and so on. Almuhareb (2006), Baroni and Lenci (2008), and Baroni et al. (2010) use the words co-occurring with a noun to approximate its most prototypical properties and correlate distributionally derived data with the properties produced by human subjects. Cimiano and Wenderoth (2007) instead focus on that subset of noun properties known in lexical semantics as qualia roles (Pustejovsky 1995), and use lexical patterns to identify, for example, the constitutive parts of a concept or its function (this is in turn analogous to the problem of relation extraction). The distributional semantics methodology also extends to more complex aspects of word meaning, addressing issues such as verb selectional preferences (Erk 2007), argument alternations (Merlo and Stevenson 2001; Joanis, Stevenson, and James 2008), event types (Zarcone and Lenci 2008), and so forth. Finally, some DSMs capture

a sort of "topical" relatedness between words: They might find, for example, a relation between *dog* and *fidelity*. Topical relatedness, addressed by DSMs based on document distributions such as LSA (Landauer and Dumais 1997) and Topic Models (Griffiths, Steyvers, and Tenenbaum 2007), is not further discussed in this article.

DSMs have found wide applications in computational lexicography, especially for automatic thesaurus construction (Grefenstette 1994; Lin 1998a; Curran and Moens 2002; Kilgarriff et al. 2004; Rapp 2004). Corpus-based semantic models have also attracted the attention of lexical semanticists as a way to provide the notion of synonymy with a more robust empirical foundation (Geeraerts 2010; Heylen et al. 2008). Moreover, DSMs for attributional and relational similarity are widely used for the semi-automatic bootstrapping or extension of terminological repositories, computational lexicons (e.g., WordNet), and ontologies (Buitelaar, Cimiano, and Magnini 2005; Lenci 2010). Innovative applications of corpus-based semantics are also being explored in linguistics, for instance in the study of semantic change (Sagi, Kaufmann, and Clark 2009), lexical variation (Peirsman and Speelman 2009), and for the analysis of multiword expressions (Alishahi and Stevenson 2008).

The wealth and variety of semantic issues that DSMs are able to tackle confirms the importance of looking at distributional data to explore meaning, as well as the maturity of this research field. However, if we looked from a distance at the whole field of DSMs we would see that, besides the general assumption shared by all models that information about the context of a word is an important key in grasping its meaning, the elements of difference overcome the commonalities. For instance, DSMs geared towards attributional similarity represent words in the contexts of other (content) words, thereby looking very different from models that represent word pairs in terms of patterns linking them. In turn, both these models differ from those used to explore concept properties or argument alternations. The typical approach in the field has been a local one, in which each semantic task (or set of closely related tasks) is treated as a separate problem, that requires its own corpus-derived model and algorithm, both optimized to achieve the best performance in a given task, but lacking generality, since they resort to task-specific distributional representations, often complemented by additional task-specific resources. As a consequence, the landscape of DSMs looks more like a jigsaw puzzle in which different parts have been completed and the whole figure starts to emerge from the fragments, but it is not clear yet how to put everything together and compose a coherent picture.

We argue that the "one semantic task, one distributional model" approach represents a great limit of the current state of the art. From a theoretical perspective, corpus-based models hold promise as large-scale simulations of how humans acquire and use conceptual and linguistic information from their environment (Landauer and Dumais 1997). However, existing DSMs lack exactly the multi-purpose nature that is a hallmark of human semantic competence. The common view in cognitive (neuro)science is that humans resort to a single semantic memory, a relatively stable long-term knowledge database, adapting the information stored there to the various tasks at hand (Murphy 2002; Rogers and McClelland 2004). The fact that DSMs need to go back to their environment (the corpus) to collect ad hoc statistics for each semantic task, and the fact that different aspects of meaning require highly different distributional representations, cast many shadows on the plausibility of DSMs as general models of semantic memory. From a practical perspective, going back to the corpus to train a different model for each application is inefficient, and it runs the risk of overfitting the model to a specific task, while losing sight of its adaptivity—a highly desirable feature for any intelligent system. Think, by contrast, of WordNet (Fellbaum 1998), a single, general purpose

network of semantic information that has been adapted to all sorts of tasks, many of them certainly not envisaged by the resource creators. We think that it is not by chance that no comparable resource has emerged from DSM development.

In this article, we want to show that a unified approach is not only a desirable goal, but it is also a feasible one. With this aim in mind, we introduce **Distributional Memory** (DM), a generalized framework for distributional semantics. Differently from other current proposals that share similar aims, we believe that the lack of generalization in corpus-based semantics stems from the choice of representing co-occurrence statistics directly as matrices—geometrical objects that model distributional data in terms of binary relations between target items (the matrix rows) and their contexts (the matrix columns). This results in the development of ad hoc models that lose sight of the fact that different semantic spaces actually rely on the same kind of underlying distributional information. DM instead represents corpus-extracted co-occurrences as a third-order tensor, a ternary geometrical object that models distributional data in terms of word–link–word tuples. Matrices are then generated from the tensor in order to perform semantic tasks in the spaces they define. Crucially, these on-demand matrices are derived from the same underlying resource (the tensor) and correspond to different "views" of the same data, extracted once and for all from a corpus. DM is tested here on what we believe to be the most varied array of semantic tasks ever addressed by a single distributional model. In all cases, we compare the performance of several DM implementations to state-of-the-art results. While some of the ad hoc models that were developed to tackle specific tasks do outperform our most successful DM implementation, the latter is never too far from the top, without any task-specific tuning. We think that the advantage of having a general model that does not need to be retrained for each new task outweighs the (often minor) performance advantage of the task-specific models.

The article is structured as follows. After framing our proposal within the general debate on co-occurrence modeling in distributional semantics (Section 2), we introduce the DM framework in Section 3 and compare it to other unified approaches in Section 4. Section 5 pertains to the specific implementations of the DM framework we will test experimentally. The experiments are reported in Section 6. Section 7 concludes by summarizing what we have achieved, and discussing the implications of these results for corpus-based distributional semantics.

2. Modeling Co-occurrence in Distributional Semantics

Corpus-based semantics aims at characterizing the meaning of linguistic expressions in terms of their distributional properties. The standard view models such properties in terms of two-way structures, that is, matrices coupling target elements (either single words or whatever other linguistic constructions we try to capture distributionally) and contexts. In fact, the formal definition of semantic space provided by Padó and Lapata (2007) is built around the notion of a matrix $\mathbf{M}_{|B|\times|T|}$, with B the set of basis elements representing the contexts used to compare the distributional similarity of the target elements T.

This binary structure is inherently suitable for approaches that represent distributional data in terms of unstructured co-occurrence relations between an element and a context. The latter can be either documents (Landauer and Dumais 1997; Griffiths, Steyvers, and Tenenbaum 2007) or lexical collocates within a certain distance from the target (Lund and Burgess 1996; Schütze 1997; Rapp 2003; Bullinaria and Levy 2007). We will refer to such models as **unstructured DSMs**, because they do not use the linguistic structure of texts to compute co-occurrences, and only record whether the target occurs

in or close to the context element, without considering the type of this relation. For instance, an unstructured DSM might derive from a sentence like *The teacher eats a red apple* that *eat* is a feature shared by *apple* and *red*, just because they appear in the same context window, without considering the fact that there is no real linguistic relation linking *eat* and *red*, besides that of linear proximity.

In **structured DSMs**, co-occurrence statistics are collected instead in the form of corpus-derived triples: typically, word pairs and the parser-extracted syntactic relation or lexico-syntactic pattern that links them, under the assumption that the surface connection between two words should cue their semantic relation (Grefenstette 1994; Lin 1998a; Curran and Moens 2002; Almuhareb and Poesio 2004; Turney 2006b; Padó and Lapata 2007; Erk and Padó 2008; Rothenhäusler and Schütze 2009). Distributional triples are also used in computational lexicography to identify the grammatical and collocational behavior of a word and to define its semantic similarity spaces. For instance, the Sketch Engine[1] builds "word sketches" consisting of triples extracted from parsed corpora and formed by two words linked by a grammatical relation (Kilgarriff et al. 2004). The number of shared triples is then used to measure the attributional similarity between word pairs.

Structured models take into account the crucial role played by syntactic structures in shaping the distributional properties of words. To qualify as context of a target item, a word must be linked to it by some (interesting) lexico-syntactic relation, which is also typically used to distinguish the type of this co-occurrence. Given the sentence *The teacher eats a red apple*, structured DSMs would not consider *eat* as a legitimate context for *red* and would distinguish the *object* relation connecting *eat* and *apple* as a different type of co-occurrence from the *modifier* relation linking *red* and *apple*. On the other hand, structured models require more preliminary corpus processing (parsing or extraction of lexico-syntactic patterns), and tend to be more sparse (because there are more triples than pairs). What little systematic comparison of the two approaches has been carried out (Padó and Lapata 2007; Rothenhäusler and Schütze 2009) suggests that structured models have a slight edge. In our experiments in Section 6.1 herein, the performance of unstructured and structured models trained on the same corpus is in general comparable. It seems safe to conclude that structured models are at least not worse than unstructured models—an important conclusion for us, as DM is built upon the structured DSM idea.

Structured DSMs extract a much richer array of distributional information from linguistic input, but they still represent it in the same way as unstructured models. The corpus-derived ternary data are mapped directly onto a two-way matrix, either by dropping one element from the tuple (Padó and Lapata 2007) or, more commonly, by concatenating two elements. The two words can be concatenated, treating the links as basis elements, in order to model relational similarity (Pantel and Pennacchiotti 2006; Turney 2006b). Alternatively, pairs formed by the link and one word are concatenated as basis elements to measure attributional similarity among the other words, treated as target elements (Grefenstette 1994; Lin 1998a; Curran and Moens 2002; Almuhareb and Poesio 2004; Rothenhäusler and Schütze 2009). In this way, typed DSMs obtain finer-grained features to compute distributional similarity, but, by couching distributional information as two-way matrices, they lose the high expressive power of corpus-derived triples. We believe that falling short of fully exploiting the potential of ternary

1 `http://www.sketchengine.co.uk`.

distributional structures is the major reason for the lack of unification in corpus-based semantics.

The debate in DSMs has so far mostly focused on the context choice—for example, lexical collocates vs. documents (Sahlgren 2006; Turney and Pantel 2010)—or on the costs and benefits of having structured contexts (Padó and Lapata 2007; Rothenhäusler and Schütze 2009). Although we see the importance of these issues, we believe that a real breakthrough in DSMs can only be achieved by overcoming the limits of current two-way models of distributional data. We propose here the alternative DM approach, in which the core geometrical structure of a distributional model is a three-way object, namely a third-order tensor. As in structured DSMs, we adopt word–link–word tuples as the most suitable way to capture distributional facts. However, we extend and generalize this assumption, by proposing that, once they are formalized as a three-way tensor, tuples can become the backbone of a unified model for distributional semantics. Different semantic spaces are then generated on demand through the independently motivated operation of tensor matricization, mapping the third-order tensor onto two-way matrices. The matricization of the tuple tensor produces both familiar spaces, similar to those commonly used for attributional or relational similarity, and other less known distributional spaces, which will yet prove useful for capturing some interesting semantic phenomena. The crucial fact is that all these different semantic spaces are now alternative views of the same underlying distributional object. Apparently unrelated semantic tasks can be addressed in terms of the same distributional memory, harvested only once from the source corpus. Thus, thanks to the tensor-based representation, distributional data can be turned into a general purpose resource for semantic modeling. As a further advantage, the third-order tensor formalization of corpus-based tuples allows distributional information to be represented in a similar way to other types of knowledge. In linguistics, cognitive science, and AI, semantic and conceptual knowledge is represented in terms of symbolic structures built around typed relations between elements, such as synsets, concepts, properties, and so forth. This is customary in lexical networks like WordNet (Fellbaum 1998), commonsense resources like ConceptNet (Liu and Singh 2004), and cognitive models of semantic memory (Rogers and McClelland 2004). The tensor representation of corpus-based distributional data promises to build new bridges between existing approaches to semantic representation that still appear distant in many respects. This may indeed contribute to the ongoing efforts to combine distributional and symbolic approaches to meaning (Clark and Pulman 2007).

3. The Distributional Memory Framework

We first introduce the notion of a weighted tuple structure, the format in which DM expects the distributional data extracted from the corpus to be arranged (and that it shares with traditional structured DSMs). We then show how a weighted tuple structure can be represented, in linear algebraic terms, as a labeled third-order tensor. Finally, we derive different semantic vector spaces from the tensor by the operation of labeled tensor matricization.

3.1 Weighted Tuple Structures

Relations among entities can be represented by ternary tuples, or triples. Let O_1 and O_2 be two sets of objects, and $R \subseteq O_1 \times O_2$ a set of relations between these objects. A triple $\langle o_1, r, o_2 \rangle$ expresses the fact that o_1 is linked to o_2 through the relation r. DM

(like previous structured DSMs) includes tuples of a particular type, namely, weighted distributional tuples that encode distributional facts in terms of typed co-occurrence relations among words. Let W_1 and W_2 be sets of strings representing content words, and L a set of strings representing syntagmatic co-occurrence links between words in a text. $T \subseteq W_1 \times L \times W_2$ is a set of corpus-derived tuples $t = \langle w_1, l, w_2 \rangle$, such that w_1 co-occurs with w_2 and l represents the type of this co-occurrence relation. For instance, the tuple $\langle$*marine*, *use*, *bomb*$\rangle$ in the toy example reported in Table 1 encodes the piece of distributional information that *marine* co-occurs with *bomb* in the corpus, and *use* specifies the type of the syntagmatic link between these words. Each tuple t has a weight, a real-valued score v_t, assigned by a scoring function $\sigma : W_1 \times L \times W_2 \rightarrow \mathbb{R}$. A **weighted tuple structure** consists of the set T_W of weighted distributional tuples $t_w = \langle t, v_t \rangle$ for all $t \in T$ and $\sigma(t) = v_t$. The σ function encapsulates all the operations performed to score the tuples, for example, by processing an input corpus with a dependency parser, counting the occurrences of tuples, and weighting the raw counts by mutual information. Because our focus is on how tuples, once they are harvested, should be represented geometrically, we gloss over the important challenges of choosing the appropriate W_1, L and W_2 string sets, as well as specifying σ.

In this article, we make the further assumption that $W_1 = W_2$. This is a natural assumption when the tuples represent (link-mediated) co-occurrences of word pairs. Moreover, we enforce an inverse link constraint such that for any link l in L, there is a k in L such that for each tuple $t_w = \langle \langle w_i, l, w_j \rangle, v_t \rangle$ in the weighted tuple structure T_W, the tuple $t_w^{-1} = \langle \langle w_j, k, w_i \rangle, v_t \rangle$ is also in T_W (we call k the inverse link of l). Again, this seems reasonable in our context: If we extract a tuple $\langle$*marine*, *use*, *gun*$\rangle$ and assign it a certain score, we might as well add the tuple $\langle$*gun*, *use*$^{-1}$, *marine*$\rangle$ with the same score. The two assumptions, combined, lead the matricization process described in Section 3.3 to generate exactly four distinct vector spaces that, as we discuss there, are needed for the semantic analyses we conduct. See Section 6.6 of Turney (2006b) for a discussion of similar assumptions. Still, it is worth emphasizing that the general formalism we are proposing, where corpus-extracted weighted tuple structures are represented as labeled tensors, does not strictly require these assumptions. For example, W_2 could be a larger set of "relata" including not only words, but also documents, morphological features, or even visual features (with appropriate links, such as, for word-document relations, *occurs-at-the-beginning-of*). The inverse link constraint might not be appropriate, for example, if we use an asymmetric association measure, or if we are only interested in one direction of certain grammatical relations. We leave the investigation of all these possibilities to further studies.

Table 1
A toy weighted tuple structure.

word	*link*	*word*	*weight*	*word*	*link*	*word*	*weight*
marine	own	bomb	40.0	sergeant	use	gun	51.9
marine	use	bomb	82.1	sergeant	own	book	8.0
marine	own	gun	85.3	sergeant	use	book	10.1
marine	use	gun	44.8	teacher	own	bomb	5.2
marine	own	book	3.2	teacher	use	bomb	7.0
marine	use	book	3.3	teacher	own	gun	9.3
sergeant	own	bomb	16.7	teacher	use	gun	4.7
sergeant	use	bomb	69.5	teacher	own	book	48.4
sergeant	own	gun	73.4	teacher	use	book	53.6

3.2 Labeled Tensors

DSMs adopting a binary model of distributional information (either unstructured models or structured models reduced to binary structures) are represented by matrices containing corpus-derived co-occurrence statistics, with rows and columns labeled by the target elements and their contexts. In DM, we formalize the weighted tuple structure as a **labeled third-order tensor**, from which semantic spaces are then derived through the operation of labeled matricization. Tensors are multi-way arrays, conventionally denoted by boldface Euler script letters: $\boldsymbol{\mathcal{X}}$ (Turney 2007; Kolda and Bader 2009). The order (or n-way) of a tensor is the number of indices needed to identify its elements. Tensors are a generalization of vectors and matrices. The entries in a vector can be denoted by a single index. Vectors are thus first-order tensors, often indicated by a bold lowercase letter: $\mathbf{v}$. The i-th element of a vector $\mathbf{v}$ is indicated by v_i. Matrices are second-order tensors, and are indicated with bold capital letters: $\mathbf{A}$. The entry (i, j) in the i-th row and j-th column of a matrix $\mathbf{A}$ is denoted by a_{ij}. An array with three indices is a third-order (or three-way) tensor. The element (i, j, k) of a third-order tensor $\boldsymbol{\mathcal{X}}$ is denoted by x_{ijk}. A convenient way to display third-order tensors is via nested tables such as Table 2, where the first index is in the header column, the second index in the first header row, and the third index in the second header row. The entry x_{321} of the tensor in the table is 7.0 and the entry x_{112} is 85.3. An index has dimensionality I if it ranges over the integers from 1 to I. The dimensionality of a third-order tensor is the product of the dimensionalities of its indices $I \times J \times K$. For example, the third-order tensor in Table 2 has dimensionality $3 \times 2 \times 3$.

If we fix the integer i as the value of the first index of a matrix $\mathbf{A}$ and take the entries corresponding to the full range of values of the other index j, we obtain a row vector (that we denote $\mathbf{a}_{i*}$). Similarly, by fixing the second index to j, we obtain the column vector $\mathbf{a}_{*j}$. Generalizing, a **fiber** is equivalent to rows and columns in higher order tensors, and it is obtained by fixing the values of all indices but one. A mode-n fiber is a fiber where only the n-th index has not been fixed. For example, in the tensor $\boldsymbol{\mathcal{X}}$ of Table 2, $\mathbf{x}_{*11} = (40.0, 16.7, 5.2)$ is a mode-1 fiber, $\mathbf{x}_{2*3} = (8.0, 10.1)$ is a mode-2 fiber, and $\mathbf{x}_{32*} = (7.0, 4.7, 53.6)$ is a mode-3 fiber.

A weighted tuple structure can be represented as a third-order tensor whose entries contain the tuple scores. As for the two-way matrices of classic DSMs, in order to make tensors linguistically meaningful we need to assign linguistic labels to the elements of the tensor indices. We define a labeled tensor $\boldsymbol{\mathcal{X}}_\lambda$ as a tensor such that for each of its indices there is a one-to-one mapping of the integers from 1 to I (the dimensionality of the index) to I distinct strings, that we call the **labels** of the index. We will refer herein to the string λ uniquely associated to index element i as the label of i, their correspondence

Table 2
A labeled third-order tensor of dimensionality $3 \times 2 \times 3$ representing the weighted tuple structure of Table 1.

	j=1:own	*j=2:use*	*j=1:own*	*j=2:use*	*j=1:own*	*j=2:use*
	k=1:bomb		*k=2:gun*		*k=3:book*	
i=1:marine	40.0	82.1	85.3	44.8	3.2	3.3
i=2:sergeant	16.7	69.5	73.4	51.9	8.0	10.1
i=3:teacher	5.2	7.0	9.3	4.7	48.4	53.6

being indicated by $i : \lambda$. A simple way to perform the mapping—the one we apply in the running example of this section—is by sorting the I items in the string set alphabetically, and mapping increasing integers from 1 to I to the sorted strings.

A weighted tuple structure T_W built from W_1, L, and W_2 can be represented by a labeled third-order tensor $\mathcal{X}_\lambda$ with its three indices labeled by W_1, L, and W_2, respectively, and such that for each weighted tuple $t \in T_W = \langle\langle w_1, l, w_2\rangle, v_t\rangle$ there is a tensor entry $(i : w_1, j : l, k : w_2) = v_t$. In other terms, a weighted tuple structure corresponds to a tensor whose indices are labeled with the string sets forming the triples, and whose entries are the tuple weights. Given the toy weighted tuple structure in Table 1, the object in Table 2 is the corresponding labeled third-order tensor.

3.3 Labeled Matricization

Matricization rearranges a higher order tensor into a matrix (Kolda 2006; Kolda and Bader 2009). The simplest case is mode-n matricization, which arranges the mode-n fibers to be the columns of the resulting $D_n \times D_j$ matrix (where D_n is the dimensionality of the n-th index, D_j is the product of the dimensionalities of the other indices). Mode-n matricization of a third-order tensor can be intuitively understood as the process of making vertical, horizontal, or depth-wise slices of a three-way object like the tensor in Table 2, and arranging these slices sequentially to obtain a matrix (a two-way object). Matricization unfolds the tensor into a matrix with the n-th index indexing the rows of the matrix and a column for each pair of elements from the other two tensor indices. For example, the mode-1 matricization of the tensor in Table 2 results in a matrix with the entries vertically arranged as they are in the table, but replacing the second and third indices with a single index ranging from 1 to 6 (cf. matrix **A** of Table 3). More explicitly, in mode-n matricization we map each tensor entry $(i_1, i_2, ..., i_N)$ to matrix entry (i_n, j), where j is computed as in Equation (1), adapted from Kolda and Bader (2009).

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} ((i_k - 1) \prod_{\substack{m=1 \\ m \neq n}}^{k-1} D_m) \tag{1}$$

For example, if we apply mode-1 matricization to the tensor of dimensionality $3 \times 2 \times 3$ in Table 2, we obtain the matrix $\mathbf{A}_{3\times 6}$ in Table 3 (ignore the labels for now). The tensor entry $x_{3,1,1}$ is mapped to the matrix cell $a_{3,1}$; $x_{3,2,3}$ is mapped to $a_{3,6}$; and $x_{1,2,2}$ is mapped to $a_{1,4}$. Observe that each column of the matrix is a mode-1 fiber of the tensor: The first column is the $\mathbf{x}_{*11}$ fiber; the second column is the $\mathbf{x}_{*21}$ fiber, and so on.

Matricization has various mathematically interesting properties and practical applications in computations involving tensors (Kolda 2006). In DM, matricization is applied to labeled tensors and it is the fundamental operation for turning the third-order tensor representing the weighted tuple structure into matrices whose row and column vector spaces correspond to the linguistic objects we want to study; that is, the outcome of matricization must be labeled matrices. Therefore, we must define an operation of labeled mode-n matricization. Recall from earlier discussion that when mode-n matricization is applied, the n-th index becomes the row index of the resulting matrix, and the corresponding labels do not need to be updated. The problem is to determine the labels of the column index of the resulting matrix. We saw that the columns of the matrix produced by mode-n matricization are the mode-n fibers of the original tensor. We must

Table 3
Labeled mode-1, mode-2, and mode-3 matricizations of the tensor in Table 2.

$\mathbf{A}_{\text{mode-1}}$	*1:⟨own, bomb⟩*	*2:⟨use, bomb⟩*	*3:⟨own, gun⟩*	*4:⟨use, gun⟩*	*5:⟨own, book⟩*	*6:⟨use, book⟩*
1:marine	40.0	82.1	85.3	44.8	3.2	3.3
2:sergeant	16.7	69.5	73.4	51.9	8.0	10.1
3:teacher	5.2	7.0	9.3	4.7	48.4	53.6

$\mathbf{B}_{\text{mode-2}}$	*1:⟨marine, bomb⟩*	*2:⟨serg., bomb⟩*	*3:⟨teacher, bomb⟩*	*4:⟨marine, gun⟩*	*5:⟨serg., gun⟩*	*6:⟨teacher, gun⟩*	*7:⟨marine, book⟩*	*8:⟨serg., book⟩*	*9:⟨teach., book⟩*
1:own	40.0	16.7	5.2	85.3	73.4	9.3	3.2	8.0	48.4
2:use	82.1	69.5	7.0	44.8	51.9	4.7	3.3	10.1	53.6

$\mathbf{C}_{\text{mode-3}}$	*1:⟨marine, own⟩*	*2:⟨marine, use⟩*	*3:⟨sergeant, own⟩*	*4:⟨sergeant, use⟩*	*5:⟨teacher, own⟩*	*6:⟨teacher, use⟩*
1:bomb	40.0	82.1	16.7	69.5	5.2	7.0
2:gun	85.3	44.8	73.4	51.9	9.3	4.7
3:book	3.2	3.3	8.0	10.1	48.4	53.6

therefore assign a proper label to mode-n tensor fibers. A mode-n fiber is obtained by fixing the values of two indices, and by taking the tensor entries corresponding to the full range of values of the third index. Thus, the natural choice for labeling a mode-n fiber is to use the pair formed by the labels of the two index elements that are fixed. Specifically, each mode-n fiber of a tensor $\boldsymbol{\mathcal{X}}_\lambda$ is labeled with the binary tuple whose elements are the labels of the corresponding fixed index elements. For instance, given the labeled tensor in Table 2, the mode-1 fiber $\mathbf{x}_{*11} = (40, 16.7, 5.2)$ is labeled with the pair ⟨*own, bomb*⟩, the mode-2 fiber $\mathbf{x}_{2*1} = (16.7, 69.5)$ is labeled with the pair ⟨*sergeant, bomb*⟩, and the mode-3 fiber $\mathbf{x}_{32*} = (7.0, 4.7, 53.6)$ is labeled with the pair ⟨*teacher, use*⟩.

Because mode-n fibers are the columns of the matrices obtained through mode-n matricization, we define the operation of **labeled mode-*n* matricization** that, given a labeled third-order tensor $\boldsymbol{\mathcal{X}}_\lambda$, maps each entry $(i_1 : \lambda_1, i_2 : \lambda_2, i_3 : \lambda_3)$ to the labeled entry $(i_n : \lambda_n, j : \lambda_j)$ such that j is obtained according to Equation (1), and λ_j is the binary tuple obtained from the triple $\langle \lambda_1, \lambda_2, \lambda_3 \rangle$ by removing λ_n. For instance, in mode-1 matricization, the entry (1:*marine*, 1:*own*, 2:*gun*) in the tensor in Table 2 is mapped onto the entry (1:*marine*, 3:⟨*own, gun*⟩). Table 3 reports the matrices **A**, **B**, and **C**, respectively, obtained by applying labeled mode-1, mode-2, and mode-3 matricization to the labeled tensor in Table 2. The columns of each matrix are labeled with pairs, according to the definition of labeled matricization we just gave. From now on, when we refer to mode-n matricization we always assume we are performing *labeled* mode-n matricization.

The rows and columns of the three matrices resulting from n-mode matricization of a third-order tensor are vectors in spaces whose dimensions are the corresponding column and row elements. Such vectors can be used to perform all standard linear algebra operations applied in vector-based semantics: Measuring the cosine of the angle between vectors, applying singular value decomposition (SVD) to the whole matrix, and so on. Under the assumption that $W_1 = W_2$ and the inverse link constraint (see Section 3.1), it follows that for each column of the matrix resulting from mode-1 matricization and labeled by $\langle l, w_2 \rangle$, there will be a column in the matrix resulting

from mode-3 matricization that is labeled by $\langle w_1, k \rangle$ (with k being the inverse link of l and $w_1 = w_2$) and that is identical to the former, except possibly for the order of the dimensions (which is irrelevant to all operations we perform on matrices and vectors, however). Similarly, for any row w_2 in the matrix resulting from mode-3 matricization, there will be an identical row w_1 in the mode-1 matricization. Therefore, given a weighted tuple structure T_W extracted from a corpus and subject to the constraints we just mentioned, by matricizing the corresponding labeled third-order tensor $\boldsymbol{\mathcal{X}}_\lambda$ we obtain the following four distinct semantic vector spaces:

word by link–word ($W_1 \times LW_2$): vectors are labeled with words w_1, and vector dimensions are labeled with tuples of type $\langle l, w_2 \rangle$;

word–word by link ($W_1W_2 \times L$): vectors are labeled with tuples of type $\langle w_1, w_2 \rangle$, and vector dimensions are labeled with links l;

word–link by word ($W_1L \times W_2$): vectors are labeled with tuples of type $\langle w_1, l \rangle$, and vector dimensions are labeled with words w_2;

link by word–word ($L \times W_1W_2$): vectors are labeled with links l and vector dimensions are labeled with tuples of type $\langle w_1, w_2 \rangle$.

Words like *marine* and *teacher* are represented in the $W_1 \times LW_2$ space by vectors whose dimensions correspond to features such as $\langle use, gun \rangle$ or $\langle own, book \rangle$. In this space, we can measure the similarity of words to each other, in order to tackle attributional similarity tasks such as synonym detection or concept categorization. The $W_1W_2 \times L$ vectors represent instead word pairs in a space whose dimensions are links, and it can be used to measure relational similarity among different pairs. For example, one could notice that the link vector of $\langle sergeant, gun \rangle$ is highly similar to that of $\langle teacher, book \rangle$. Crucially, as can be seen in Table 3, the corpus-derived scores that populate the vectors in these two spaces are exactly the same, just arranged in different ways. In DM, attributional and relational similarity spaces are different views of the same underlying tuple structure.

The other two distinct spaces generated by tensor matricization look less familiar, and yet we argue that they allow us to subsume under the same general DM framework other interesting semantic phenomena. We will show in Section 6.3 how the $W_1L \times W_2$ space can be used to capture different verb classes based on the argument alternations they display. For instance, this space can be used to find out that the object slot of *kill* is more similar to the subject slot of *die* than to the subject slot of *kill* (and, generalizing from similar observations, that the subject slot of *die* is a theme rather than an agent). The $L \times W_1W_2$ space displays similarities among links. The usefulness of this will of course depend on what the links are. We will illustrate in Section 6.4 one function of this space, namely, to perform feature selection, picking links that can then be used to determine a meaningful subspace of the $W_1W_2 \times L$ space.

Direct matricization is just one of the possible uses we can make of the labeled tensor. In Section 6.5 we illustrate another use of the tensor formalism by performing smoothing through tensor decomposition. Other possibilities, such as graph-based algorithms operating directly on the graph defined by the tensor (Baroni and Lenci 2009), or deriving unstructured semantic spaces from the tensor by removing one of the indices, are left to future work.

Before we move on, it is worth emphasizing that, from a computational point of view, there is virtually no additional cost in the tensor approach, with respect to traditional structured DSMs. The labeled tensor is nothing other than a formalization of

distributional data extracted in the word–link–word–score format, which is customary in many structured DSMs. Labeled matricization can then simply be obtained by concatenating two elements in the original triple to build the corresponding matrix—again, a common step in building a structured DSM. In spite of being cost-free in terms of implementation, the mathematical formalism of labeled tensors highlights the common core shared by different views of the semantic space, thereby making distributional semantics more general.

4. Related Work

As will be clear in the next sections, the ways in which we tackle specific tasks are, by themselves, mostly not original. The main element of novelty is the fact that methods originally developed to resort to ad hoc distributional spaces are now adapted to fit into the unified DM framework. We will point out connections to related research specific to the various tasks in the sections devoted to describing their reinterpretation in DM. We omit discussion of our own work that the DM framework is an extension and generalization of Baroni et al. (2010) and Baroni and Lenci (2009). Instead, we briefly discuss two other studies that explicitly advocate a uniform approach to corpus-based semantic tasks, and one article that, like us, proposes a tensor-based formalization of corpus-extracted triples. See Turney and Pantel (2010) for a very recent general survey of DSMs.

Padó and Lapata (2007), partly inspired by Lowe (2001), have proposed an interesting general formalization of DSMs. In their approach, a corpus-based semantic model is characterized by (1) a set of functions to extract statistics from the corpus, (2) construction of the basis-by-target-elements co-occurrence matrix, and (3) a similarity function operating on the matrix. Our focus is entirely on the second aspect. A DM, according to the characterization in Section 3, is a labeled tensor based on a source weighted tuple structure and coupled with matricization operations. How the tuple structure was built (corpus extraction methods, association measures, etc.) is not part of the DM formalization. At the other end, DM provides sets of vectors in different vector spaces, but it is agnostic about how they are used (measuring similarity via cosines or other measures, reducing the matrices with SVD, etc.). Of course, much of the interesting progress in distributional semantics will occur at the two ends of our tensor, with better tuple extraction and weighting techniques on one side, and better matrix manipulation and similarity measurement on the other. As long as the former operations result in data that can be arranged into a weighted tuple structure, and the latter procedures act on vectors, such innovations fit into the DM framework and can be used to improve performance on tasks defined on any space derivable from the DM tensor.

Whereas the model proposed by Padó and Lapata (2007) is designed only to address tasks involving the measurement of the attributional similarity between words, Turney (2008) shares with DM the goal of unifying attributional and relational similarity under the same distributional model. He observes that tasks that are traditionally solved with an attributional similarity approach can be recast as relational similarity tasks. Instead of determining whether two words are, for example, synonymous by looking at the features they share, we can learn what the typical patterns are that connect synonym pairs when they co-occur (*also known as*, *sometimes called*, etc.), and make a decision about a potential synonym pair based on their occurrence in similar contexts. Given a list of pairs instantiating an arbitrary relation, Turney's PairClass algorithm extracts patterns that are correlated with the relation, and can be used to discover new

pairs instantiating it. Turney tests his system in a variety of tasks (TOEFL synonyms; SAT analogies; distinguishing synonyms and antonyms; distinguishing pairs that are semantically similar, associated, or both), obtaining good results across the board.

In the DM approach, we collect one set of statistics from the corpus, and then exploit different views of the extracted data and different algorithms to tackle different tasks. Turney, on the contrary, uses a single generic algorithm, but must go back to the corpus to obtain new training data for each new task. We compare DM with some of Turney's results in Section 6 but, independently of performance, we find the DM approach more appealing. As corpora grow in size and are enriched with further levels of annotation, extracting ad hoc data from them becomes a very time-consuming operation. Although we did not carry out any systematic experiments, we observe that the extraction of tuple counts from (already POS-tagged and parsed) corpora in order to train our sample DM models took days, whereas even the most time-consuming operations to adapt DM to a task took on the order of 1 to 2 hours on the same machines (task-specific training is also needed in PairClass, anyway). Similar considerations apply to space: Compressed, our source corpora take about 21 GB, our best DM tensor (TypeDM) 1.1 GB (and optimized sparse tensor representations could bring this quantity down drastically, if the need arises). Perhaps more importantly, extracting features from the corpus requires a considerable amount of NLP know-how (to pre-process the corpus appropriately, to navigate a dependency tree, etc.), whereas the DM representation of distributional data as weighted triples is more akin to other standard knowledge representation formats based on typed relations, which are familiar to most computer and cognitive scientists. Thus, a trained DM can become a general-purpose resource and be used by researchers beyond the realms of the NLP community, whereas applying PairClass requires a good understanding of various aspects of computational linguistics. This severely limits its interdisciplinary appeal.

At a more abstract level, DM and PairClass differ in the basic strategy with which unification in distributional semantics is pursued. Turney's approach amounts to picking a task (identifying pairs expressing the same relation) and reinterpreting other tasks as its particular instances. Thus, attributional and relational similarity are unified by considering the former as a subtype of the latter. Conversely, DM assumes that each semantic task may keep its specificity, and unification is achieved by designing a sufficiently general distributional structure, populating a specific instance of the structure, and generating semantic spaces on demand from the latter. This way, DM is able to address a wider range of semantic tasks than Turney's model. For instance, language is full of productive semantic phenomena, such as the selectional preferences of verbs with respect to unseen arguments (*eating topinambur* vs. *eating sympathy*). Predicting the plausibility of unseen pairs cannot, by definition, be tackled by the current version of PairClass, which will have to be expanded to deal with such cases, perhaps adopting ideas similar to those we present (that are, in turn, inspired by Turney's own work on attributional and relational similarity). A first step in this direction, within a framework similar to Turney's, was taken by Herdağdelen and Baroni (2009).

Turney (2007) explicitly formalizes the set of corpus-extracted word–link–word triples as a tensor, and was our primary source of inspiration in formalizing DM in these terms. The focus of Turney's article, however, is on dimensionality reduction techniques applied to tensors, and the application to corpora is only briefly discussed. Moreover, Turney only derives the $W_1 \times LW_2$ space from the tensor, and does not discuss the possibility of using the tensor-based formalization to unify different views of semantic data, which is instead our main point. The higher-order tensor dimensionality reduction techniques tested on language data by Turney (2007) and Van de Cruys (2009) can be

applied to the DM tensors before matricization. We present a pilot study in this direction in Section 6.5.

5. Implementing DM

5.1 Extraction of Weighted Tuple Structures from Corpora

In order to make our proposal concrete, we experiment with three different DM models, corresponding to different ways to construct the underlying weighted tuple structure (Section 3.1). All models are based on the natural idea of extracting word–link–word tuples from a dependency parse of a corpus, but this is not a requirement for DM: The links could for example be based on frequent *n*-grams as in Turney (2006b) and Baroni et al. (2010), or even on very different kinds of relation, such as co-occurring within the same document.

The current models are trained on the concatenation of (1) the Web-derived ukWaC corpus,[2] about 1.915 billion tokens (here and subsequently, counting only strings that are entirely made of alphabetic characters); (2) a mid-2009 dump of the English Wikipedia,[3] about 820 million tokens; and (3) the British National Corpus,[4] about 95 million tokens. The resulting concatenated corpus was tokenized, POS-tagged, and lemmatized with the TreeTagger[5] and dependency-parsed with the MaltParser.[6] It contains about 2.83 billion tokens. The ukWaC and Wikipedia sections can be freely downloaded, with full annotation, from the ukWaC corpus site.

For all our models, the label sets $W_1 = W_2$ contain 30,693 lemmas (20,410 nouns, 5,026 verbs, and 5,257 adjectives). These terms were selected based on their frequency in the corpus (they are approximately the top 20,000 most frequent nouns and top 5,000 most frequent verbs and adjectives), augmenting the list with lemmas that we found in various standard test sets, such as the TOEFL and SAT lists. In all models, the words are stored in POS-suffixed lemma form. The weighted tuple structures differ for the choice of links in L and/or for the scoring function σ.

DepDM. Our first DM model relies on the classic intuition that dependency paths are a good approximation to semantic relations between words (Grefenstette 1994; Curran and Moens 2002; Padó and Lapata 2007; Rothenhäusler and Schütze 2009). DepDM is also the model with the least degree of link lexicalization among the three DM instances we have built (its only lexicalized links are prepositions). L_{DepDM} includes the following noun–verb, noun–noun, and adjective–noun links (in order to select more reliable dependencies and filter out possible parsing errors, dependencies between words with more than five intervening items were discarded):

sbj_intr: subject of a verb that has no direct object: *The teacher is singing* → ⟨*teacher*, sbj_intr, *sing*⟩; *The soldier talked with his sergeant* → ⟨*soldier*, sbj_intr, *talk*⟩;

sbj_tr: subject of a verb that occurs with a direct object: *The soldier is reading a book* → ⟨*soldier*, sbj_tr, *read*⟩;

2 http://wacky.sslmit.unibo.it/.
3 http://en.wikipedia.org/wiki/Wikipedia:Database_download.
4 http://www.natcorp.ox.ac.uk.
5 http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.
6 http://w3.msi.vxu.se/~nivre/research/MaltParser.html.

obj: direct object: *The soldier is reading a book* → ⟨*book*, obj, *read*⟩;

iobj: indirect object in a double object construction: *The soldier gave the woman a book* → ⟨*woman*, iobj, *give*⟩;

nmod: noun modifier: *good teacher* → ⟨*good*, nmod, *teacher*⟩; *school teacher* → ⟨*school*, nmod, *teacher*⟩;

coord: noun coordination: *teachers and soldiers* → ⟨*teacher*, coord, *soldier*⟩;

prd: predicate noun: *The soldier became sergeant* → ⟨*sergeant*, prd, *become*⟩;

verb: an underspecified link between a subject noun and a complement noun of the same verb: *The soldier talked with his sergeant* → ⟨*soldier*, verb, *sergeant*⟩; *The soldier is reading a book* → ⟨*soldier*, verb, *book*⟩;

preposition: every preposition linking the noun head of a prepositional phrase to its noun or verb head (a different link for each preposition): *I saw a soldier with the gun* → ⟨*gun*, with, *soldier*⟩; *The soldier talked with his sergeant* → ⟨*sergeant*, with, *talk*⟩.

For each link, we also extract its inverse (this holds for all our DM models). For example, there is a sbj_intr^{-1} link between an intransitive verb and its subject: ⟨*talk*, sbj_intr^{-1}, *soldier*⟩. The cardinality of L_{DepDM} is 796, including direct and inverse links.

The weights assigned to the tuples by the scoring function σ are given by **Local Mutual Information** (LMI) computed on the raw corpus-derived word–link–word co-occurrence counts. Given the co-occurrence count O_{ijk} of three elements of interest (in our case, the first word, the link, and the second word), and the corresponding expected count under independence E_{ijk}, $\text{LMI} = O_{ijk} \log \frac{O_{ijk}}{E_{ijk}}$. LMI is an approximation to the log-likelihood ratio measure that has been shown to be a very effective weighting scheme for sparse frequency counts (Dunning 1993; Padó and Lapata 2007). The measure can also be interpreted as the dominant term of average MI or as a heuristic variant of pointwise MI to avoid its bias towards overestimating the significance of low frequency events, and it is nearly identical to the Poisson–Stirling measure (Evert 2005). LMI has considerable computational advantages in cases like ours, in which we measure the association of three elements, because it does not require keeping track of the full $2 \times 2 \times 2$ contingency table, which is the case for the log-likelihood ratio. Following standard practice (Bullinaria and Levy 2007), negative weights (cases where the observed value is lower than the expected value) are raised to 0. The number of non-zero tuples in the DepDM tensor is about 110M, including tuples with direct links and their inverses. DepDM is a $30,693 \times 796 \times 30,693$ tensor with density 0.0149% (the proportion of non-zero entries in the tensor).

LexDM. The second model is inspired by the idea that the lexical material connecting two words is very informative about their relation (Hearst 1992; Pantel and Pennacchiotti 2006; Turney 2006b). L_{LexDM} contains complex links, each with the structure pattern+*suffix*. The suffix is in turn formed by two substrings separated by a +, each respectively encoding the following features of w_1 and w_2: their POS and morphological features (number for nouns, number and tense for verbs); the presence of an article (further specified with its definiteness value) and of adjectives for nouns; the presence of adverbs for adjectives; and the presence of adverbs, modals, and auxiliaries for verbs, together with their diatheses (for passive only). If the adjective (adverb) modifying w_1 or w_2 belongs to a list of 10 (250) high frequency adjectives (adverbs), the suffix

string contains the adjective (adverb) itself, otherwise only its POS. For instance, from the sentence *The tall soldier has already shot* we extract the tuple ⟨*soldier,* sbj_intr+*n-the-j+vn-aux-already, shoot*⟩. Its complex link contains the pattern sbj_intr and the suffix *n-the-j+vn-aux-already*. The suffix substring *n-the-j* encodes the information that w_1 is a singular noun (*n*), is definite (*the*), and has an adjective (*j*) that does not belong to the list of high frequency adjectives. The substring *vn-aux-already* specifies that w_2 is a past-participle (*vn*), has an auxiliary (*aux*), and is modified by *already*, belonging to the pre-selected list of high frequency adverbs. The patterns in the LexDM links include:

L_{DepDM} : every DepDM link is a potential pattern of a LexDM link: *The soldier has shot* → ⟨*soldier,* sbj_intr+*n-the+vn-aux, shoot*⟩;

verb: if the verb link between a subject noun and a complement noun belongs to a list of 52 high frequency verbs, the underspecified verb link of DepDM is replaced by the verb itself: *The soldier used a gun* → ⟨*soldier,* use+*n-the+n-a, gun*⟩; *The soldier read the yellow book* → ⟨*soldier,* verb+*n-the+n-the-j, book*⟩;

is: copulative structures with an adjectival predicate: *The soldier is tall* → ⟨*tall,* is+*j+n-the, soldier*⟩;

preposition–link_noun–preposition: this schema captures connecting expressions such as *of a number of*, *in a kind of*; *link_noun* is one of 48 semi-manually selected nouns such as *number*, *variety*, or *kind*; *the arrival of a number of soldiers* → ⟨*soldier,* of-number-of+*ns+n-the, arrival*⟩;

attribute_noun: one of 127 nouns extracted from WordNet and expressing attributes of concepts, such as *size*, *color*, or *height*. This pattern connects adjectives and nouns that occur in the templates *(the) attribute_noun of (a|the) NOUN is ADJ* (Almuhareb and Poesio 2004) and *(a|the) ADJ attribute_noun of NOUN* (Veale and Hao 2008): *the color of strawberries is red* → ⟨*red,* color+*j+ns, strawberry*⟩; *the autumnal color of the forest* → ⟨*autumnal,* color+*j+n-the, forest*⟩;

as_adj_as: this pattern links an adjective and a noun that match the template *as ADJ as (a|the) NOUN* (Veale and Hao 2008): *as sharp as a knife* → ⟨*sharp,* as_adj_as+*j+n-a, knife*⟩;

such_as: links two nouns occurring in the templates *NOUN such as NOUN* and *such NOUN as NOUN* (Hearst 1992, 1998): *animals such as cats* → ⟨*animal,* such_as+*ns+ns, cat*⟩; *such vehicles as cars* → ⟨*vehicle,* such_as+*ns+ns, car*⟩.

LexDM links have a double degree of lexicalization. First, the suffixes encode a wide array of surface features of the tuple elements. Secondly, the link patterns themselves, besides including standard syntactic relations (such as direct object or coordination), extend to lexicalized dependency relations (specific verbs) and lexico-syntactic shallow templates. The latter include patterns adopted in the literature to extract specific pieces of semantic knowledge. For instance, *NOUN such as NOUN* and *such NOUN as NOUN* were first proposed by Hearst (1992) as highly reliable patterns for hypernym identification, whereas *(the) attribute_noun of (a|the) NOUN is ADJ* and *(a|the) ADJ attribute_noun of NOUN* were successfully used to identify typical values of concept attributes (Almuhareb and Poesio 2004; Veale and Hao 2008). Therefore, the LexDM distributional memory is a repository of partially heterogeneous types of corpus-derived information, differing in their level of abstractness, which ranges from fairly abstract syntactic relations to shallow lexicalized patterns. L_{LexDM} contains 3,352,148 links, including inverses.

The scoring function σ is the same as that in DepDM, and the number of non-zero tuples is about 355M, including direct and inverse links. LexDM is a 30,693 × 3,352,148 × 30,693 tensor with density 0.00001%.

TypeDM. This model is based on the idea, motivated and tested by Baroni et al. (2010)—but see also Davidov and Rappoport (2008a, 2008b) for a related method—that what matters is not so much the frequency of a link, but the variety of surface forms that express it. For example, if we just look at frequency of co-occurrence (or strength of association), the triple ⟨*fat*, *of*$^{-1}$, *land*⟩ (a figurative expression) is much more common than the semantically more informative ⟨*fat*, *of*$^{-1}$, *animal*⟩. However, if we count the different surface realizations of the former pattern in our corpus, we find that there are only three of them (*the fat of the land*, *the fat of the ADJ land*, and *the ADJ fat of the land*), whereas ⟨*fat*, *of*$^{-1}$, *animal*⟩ has nine distinct realizations (*a fat of the animal*, *the fat of the animal*, *fats of animal*, *fats of the animal*, *fats of the animals*, *ADJ fats of the animal*, and *the fats of the animal*). TypeDM formalizes this intuition by adopting as links the patterns inside the LexDM links, while the suffixes of these patterns are used to count their number of distinct surface realizations. We call the model TypeDM because it counts *types* of realizations, not tokens. For instance, the two LexDM links of^{-1}*+n-a+n-the* and of^{-1}*+ns-j+n-the* are counted as two occurrences of the same TypeDM link of^{-1}, corresponding to the pattern in the two original links.

The scoring function σ computes LMI not on the raw word–link–word co-occurrence counts, but on the number of distinct suffix types displayed by a link when it co-occurs with the relevant words. For instance, a TypeDM link derived from a LexDM pattern that occurs with nine different suffix types in the corpus is assigned a frequency of 9 for the purpose of the computation of LMI. The distinct TypeDM links are 25,336. The number of non-zero tuples in the TypeDM tensor is about 130M, including direct and inverse links. TypeDM is a $30,693 \times 25,336 \times 30,693$ tensor with density 0.0005%.

To sum up, the three DM instance models herein differ in the degree of lexicalization of the link set, and/or in the scoring function. LexDM is a heavily lexicalized model, contrasting with DepDM, which has a minimum degree of lexicalization, and consequently the smallest set of links. TypeDM represents a sort of middle level both for the kind and the number of links. These consist of syntactic and lexicalized patterns, as in LexDM. The lexical information encoded in the LexDM suffixes, however, is not used to generate different links, but to implement a different counting scheme as part of a different scoring function.

A weighted tuple structure (equivalently: a labeled DM tensor) is intended as a long-term semantic resource that can be used in different projects for different tasks, analogously to traditional hand-coded resources such as WordNet. Coherent with this approach, we make our best DM model (TypeDM) publicly available from `http://clic.cimec.unitn.it/dm`. The site also contains a set of Perl scripts that perform the basic operations on the tensor and its derived vectors we are about to describe.

5.2 Semantic Vector Manipulation

The DM framework provides, via matricization, a set of matrices with associated labeled row and column vectors. These labeled matrices can simply be derived from the tuple tensor by concatenating two elements in the original triples. Any operation that can be performed on the resulting matrices and that might help in tackling a semantic task is fair game. However, in the experiments reported in this article we will work with a limited number of simple operations that are well-motivated in terms of the

geometric framework we adopt, and suffice to face all the tasks we will deal with (the decomposition techniques explored in Section 6.5 are briefly introduced there).

Vector length and normalization. The length of a vector $\mathbf{v}$ with dimensions $v_1, v_2, \ldots, v_n$ is:

$$||\mathbf{v}|| = \sqrt{\sum_{i=1}^{i=n} v_i^2}$$

A vector is normalized to have length 1 by dividing each dimension by the original vector length.

Cosine. We measure the similarity of two vectors $\mathbf{x}$ and $\mathbf{y}$ by the cosine of the angle they form:

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{i=n} x_i y_i}{||\mathbf{x}||||\mathbf{y}||}$$

The cosine ranges from ± 1 for vectors pointing in the same direction to 0 for orthogonal vectors. Other similarity measures, such as Lin's measure (Lin 1998b), work better than the cosine in some tasks (Curran and Moens 2002; Padó and Lapata 2007). However, the cosine is the most natural similarity measure in the geometric formalism we are adopting, and we stick to it as the default approach to measuring similarity.

Vector sum. Two or more vectors are summed in the obvious way, by adding their values on each dimension. We always normalize the vectors before summing. The resulting vector points in the same direction as the average of the summed normalized vectors. We refer to it as the *centroid* of the vectors.

Projection onto a subspace. It is sometimes useful to measure length or compare vectors by taking only some of their dimensions into account. For example, one way to find nouns that are typical objects of the verb *to sing* is to measure the length of nouns in a $W_1 \times LW_2$ subspace in which only dimensions such as $\langle obj, sing \rangle$ have non-0 values. We project a vector onto a subspace of this kind through multiplication of the vector by a square diagonal matrix with 1s in the diagonal cells corresponding to the dimensions we want to preserve and 0s elsewhere. A matrix of this sort performs an orthogonal projection of the vector it multiplies (Meyer 2000, chapter 5).

6. Semantic Experiments with the DM Spaces

As we saw in Section 3, labeled matricization generates four distinct semantic spaces from the third-order tensor. For each space, we have selected a set of semantic experiments that we model by applying some combination of the vector manipulation operations of Section 5.2. The experiments correspond to key semantic tasks in computational linguistics and/or cognitive science, typically addressed by distinct DSMs so far. We have also aimed at maximizing the variety of aspects of meaning covered by the experiments, ranging from synonymy detection to argument structure and concept properties, and encompassing all the major lexical classes. Both these facts support the view of DM as a generalized model that is able to overtake state-of-the-art DSMs in the number and types of semantic issues addressed, while being competitive in each specific task.

The choice of the DM semantic space to tackle a particular task is essentially based on the "naturalness" with which the task can be modeled in that space. However, alternatives are conceivable, both with respect to space selection, and to the operations performed on the space. For instance, Turney (2008) models synonymy detection with a DSM that closely resembles our $W_1W_2{\times}L$ space, whereas we tackle this task under the more standard $W_1{\times}LW_2$ view. It is an open question whether there are principled ways to select the optimal space configuration for a given semantic task. In this article, we limit ourselves to proving that each space derived through tensor matricization is semantically interesting in the sense that it provides the proper ground to address some semantic task.

Feature selection/reweighting and dimensionality reduction have been shown to improve DSM performance. For instance, the feature bootstrapping method proposed by Zhitomirsky-Geffet and Dagan (2009) boosts the precision of a DSM in lexical entailment recognition. Even if these methods can be applied to DM as well, we did not use them in our experiments. The results presented subsequently should be regarded as a "baseline" performance that could be enhanced in future work by exploring various task-specific parameters (we will come back in the conclusion to the role of parameter tuning in DM). This is consistent with our current aim of focusing on the generality and adaptivity of DM, rather than on task-specific optimization. As a first, important step in this latter direction, however, we conclude the empirical evaluation in Section 6.5 by replicating one experiment using tensor-decomposition-based smoothing, a form of optimization that can only be performed within the tensor-based approach to DSMs.

In order to maximize coverage of the experimental test sets, they are pre-processed with a mixture of manual and heuristic procedures to assign a POS to the words they contain, lemmatize, convert some multiword forms to single words, and turn some adverbs into adjectives (our models do not contain multiwords or adverbs). Nevertheless, some words (or word pairs) are unrecoverable, and in such cases we make a random guess (in cases where we do not have full coverage of a data set, the reported results are averages across repeated experiments, to account for the variability in random guesses).

In many of the experiments herein, DM is not only compared to the results available in the literature, but also to our implementation of state-of-the-art DSMs. These alternative models have been trained on the same corpus (with the same linguistic preprocessing) used to build the DM tuple tensors. This way, we aim at achieving a fairer comparison with alternative approaches in distributional semantics, abstracting away from the effects induced by differences in the training data.

Most experiments report global (micro-averaged) test set accuracy (alone, or combined with other measures) to assess the performance of the algorithms. The number of correctly classified items among all test elements can be seen as a binomially distributed random variable, and we follow the ACL Wiki state-of-the-art site[7] in reporting also Clopper–Pearson binomial 95% confidence intervals around the accuracies (binomial intervals and other statistical quantities were computed using the R package;[8] where no further references are given, we used the standard R functions for the relevant analysis). The binomial confidence intervals give a sense of the spread of plausible population values around the test-set-based point estimates of accuracy. Where appropriate and interesting, we compare the accuracy of two specific models statistically with an exact Fisher test on the contingency table of correct and wrong responses given by the two

7 http://aclweb.org/aclwiki/index.php?title=State_Of_The_Art.
8 http://www.r-project.org/.

models. This approach to significance testing is problematic in many respects, the most important being that we ignore dependencies in correct and wrong counts due to the fact that the algorithms are evaluated on the same test set (Dietterich 1998). More appropriate tests, however, would require access to the fully itemized results from the compared algorithms, whereas in most cases we only know the point estimate reported in the earlier literature. For similar reasons, we do not make significance claims regarding other performance measures, such as macro-averaged F. Other forms of statistical analysis of the results are introduced herein when they are used; they are mostly limited to the models for which we have full access to the results. Note that we are interested in whether DM performance is overall within state-of-the-art range, and not on making precise claims about the models it outperforms. In this respect, we think that our general results are clear even where they are not supported by statistical inference, or interpretation of the latter is problematic.

6.1 The $W_1 \times LW_2$ Space

The vectors of this space are labeled with words w_1 (rows of matrix $\mathbf{A}_{\text{mode-1}}$ in Table 3), and their dimensions are labeled with binary tuples of type $\langle l, w_2 \rangle$ (columns of the same matrix). The dimensions represent the attributes of words in terms of lexico-syntactic relations with lexical collocates, such as $\langle$*sbj_intr, read*$\rangle$, or $\langle$*use, gun*$\rangle$. Consistently, all the semantic tasks that we address with this space involve the measurement of the attributional similarity between words.

The $W_1 \times LW_2$ matrix is a structured semantic space similar to those used by Curran and Moens (2002), Grefenstette (1994), and Lin (1998a), among others. To test if the use of links detracts from performance on attributional similarity tasks, we trained on our concatenated corpus two alternative models—Win and DV—whose features only include lexical collocates of the target. Win is an unstructured DSM that does not rely on syntactic structure to select the collocates, but just on their linear proximity to the targets (Lund and Burgess 1996; Schütze 1997; Bullinaria and Levy 2007, and many others). Its matrix is based on co-occurrences of the same 30K words we used for the other models within a window of maximally five content words before or after the target. DV is an implementation of the Dependency Vectors approach of Padó and Lapata (2007). It is a structured DSM, but dependency paths are used to pick collocates, without being part of the attributes. The DV model is obtained from the same co-occurrence data as DepDM (thus, relying on the dependency paths we picked, not the ones originally selected by Padó and Lapata for their tests). Frequencies are summed across dependency path links for word–link–word triples with the same first and second words. Suppose that *soldier* and *gun* occur in the tuples $\langle$*soldier, have, gun*$\rangle$ (frequency 3) and $\langle$*soldier, use, gun*$\rangle$ (frequency 37). In DepDM, this results in two features for *soldier*: $\langle$*have, gun*$\rangle$ and $\langle$*use, gun*$\rangle$. In DV, we would derive a single *gun* feature with frequency 40. As for the DM models, the Win and DV counts are converted to LMI weights, and negative LMI values are raised to 0. Win is a 30,693 $\times$ 30,693 matrix with about 110 million non-zero entries (density: 11.5%). DV is a 30,693 $\times$ 30,693 matrix with about 38 million non-zero values (density: 4%).

6.1.1 Similarity Judgments. Our first challenge comes from the classic data set of Rubenstein and Goodenough (1965), consisting of 65 noun pairs rated by 51 subjects on a 0–4 similarity scale. The average rating for each pair is taken as an estimate of the perceived similarity between the two words (e.g., *car–automobile*: 3.9, *cord–smile*: 0.0). Following the earlier literature, we use Pearson's r to evaluate how well the cosines

in the $W_1 \times LW_2$ space between the nouns in each pair correlate with the ratings. The results (expressed in terms of percentage correlations) are presented in Table 4, which also reports state-of-the-art performance levels of corpus-based systems from the literature (the correlation of all systems with the ratings is very significantly above chance, according to a two-tailed t-test for Pearson correlation coefficients; $df = 63$, $p < 0.0001$ for all systems).

One of the DM models, namely TypeDM, does very well on this task, outperformed only by DoubleCheck, an unstructured system that relies on Web queries (and thus on a much larger corpus) and for which we report the best result across parameter settings. We also report the best results from a range of experiments with different models and parameter settings from Herdağdelen, Erk, and Baroni (2009) (whose corpus is about half the size of ours) and Padó and Lapata (2007) (who use a much smaller corpus). For the latter, we also report the best result they obtain when using cosine as the similarity measure (cosDV-07). Overall, the TypeDM result is in line with the state of the art, given the size of the input corpus, and the fact that we did not perform any tuning. Following Padó, Padó, and Erk (2007) we used the approximate test proposed by Raghunathan (2003) to compare the correlations with the human ratings of sets of models (this is only possible for the models we developed, as the test requires computation of correlation coefficients across models). The test suggests that the difference in correlation with human ratings between TypeDM and our second best model, Win, is significant ($Q = 4.55$, $df = 0.23$, $p < 0.01$). On the other hand, there is no significant difference across Win, DepDM, DV and LexDM ($Q = 1.02$, $df = 1.80$, $p = 0.55$).

6.1.2 Synonym Detection. The previous experiment assessed how the models can simulate quantitative similarity ratings. The classic TOEFL synonym detection task focuses on the high end of the similarity scale, asking the models to make a discrete decision about which word is the synonym from a set of candidates. The data set, introduced to computational linguistics by Landauer and Dumais (1997), consists of 80 multiple-choice questions, each made of a target word (a noun, verb, adjective, or adverb) and four candidates. For example, given the target *levied*, the candidates are *imposed*, *believed*, *requested*, *correlated*, the first one being the correct choice. Our algorithms pick the candidate with the highest cosine to the target item as their guess of the right synonym.

Table 5 reports results (percentage accuracies) on the TOEFL set for our models as well as the best model of Herdağdelen and Baroni (2009) and the corpus-based models from the ACL Wiki TOEFL state-of-the-art table (we do not include those models from the Wiki that resort to other knowledge sources, such as WordNet or a thesaurus). The claims to follow about the relative performance of the models must be interpreted cautiously, in light of the spread of the confidence intervals: It suffices to note that,

Table 4
Percentage Pearson correlation with the Rubenstein and Goodenough (1965) similarity ratings.

model	*r*	*model*	*r*	*model*	*r*
DoubleCheck[1]	85	Win	65	DV	57
TypeDM	82	DV-07[3]	62	LexDM	53
SVD-09[2]	80	DepDM	57	cosDV-07[3]	47

Model sources: [1]Chen, Lin, and Wei (2006); [2]Herdağdelen, Erk, and Baroni (2009); [3]Padó and Lapata (2007).

Table 5
Percentage accuracy in TOEFL synonym detection with 95% binomial confidence intervals (CI).

model	*accuracy*	*95% CI*	*model*	*accuracy*	*95% CI*
LSA-03[1]	92.50	84.39–97.20	DepDM	75.01	64.06–84.01
GLSA[2]	86.25	76.73–92.93	LexDM	74.37	63.39–83.49
PPMIC[3]	85.00	75.26–92.00	PMI-IR-01[8]	73.75	62.72–82.96
CWO[4]	82.55	72.38–90.09	DV-07[9]	73.00	62.72–82.96
PMI-IR-03[5]	81.25	70.97–89.11	Win	69.37	58.07–79.20
BagPack[6]	80.00	69.56–88.11	Human[10]	64.50	53.01–74.88
DV	76.87	66.10–85.57	LSA-97[10]	64.38	52.90–74.80
TypeDM	76.87	66.10–85.57	Random	25.00	15.99–35.94
PairClass[7]	76.25	65.42–85.05			

Model sources: [1]Rapp (2003); [2]Matveeva et al. (2005); [3]Bullinaria and Levy (2007); [4]Ruiz-Casado, Alfonseca, and Castells (2005); [5]Terra and Clarke (2003); [6]Herdağdelen and Baroni (2009); [7]Turney (2008); [8]Turney (2001); [9]Padó and Lapata (2007); [10]Landauer and Dumais (1997).

according to a Fisher test, the difference between the second-best model, GLSA, and the twelfth model, PMI-IR-01, is not significant at the $\alpha = .05$ level ($p = 0.07$). The difference between the bottom model, LSA-97, and random guessing is, on the other hand, highly significant ($p < .00001$).

The best DM model is again TypeDM, which also outperforms Turney's unified PairClass approach (supervised, and relying on a much larger corpus), as well as his Web-statistics based PMI-IR-01 model. TypeDM does better than the best Padó and Lapata model (DV-07), and comparably to our DV implementation. Its accuracy is more than 10% higher than the average human test taker and the classic LSA model (LSA-97). Among the approaches that outperform TypeDM, BagPack is supervised, and CWO and PMI-IR-03 rely on much larger corpora. This leaves us with three unsupervised (and unstructured) models from the literature that outperform TypeDM while being trained on comparable or smaller corpora: LSA-03, GLSA, and PPMIC. In all three cases, the authors show that parameter tuning is beneficial in attaining the reported best performance. Further work should investigate how we could improve TypeDM by exploring various parameter settings (many of which do not require going back to the corpus: feature selection and reweighting, SVD, etc.).

6.1.3 Noun Categorization. Humans are able to group words into classes or categories depending on their meaning similarities. Categorization tasks play a prominent role in cognitive research on concepts and meaning, as a probe into the semantic organization of the lexicon and the ability to arrange concepts hierarchically into taxonomies (Murphy 2002). Research in corpus-based semantics has always been interested in investigating whether distributional (attributional) similarity could be used to group words into semantically coherent categories. From the computational point of view, this is a particularly crucial issue because it concerns the possibility of using distributional information to assign a semantic class or type to words. Categorization requires (at least in current settings) a discrete decision, as in the TOEFL task, but it is based on detecting not only synonyms but also less strictly related words that stand in a coordinate/co-hyponym relation. We focus here on noun categorization, which we operationalize as a clustering task. Distributional categorization has been investigated for other POS as well, most notably verbs (Merlo and Stevenson 2001; Schulte im Walde 2006). However,

verb classifications are notoriously more controversial than nominal ones, and deeply interact with argument structure properties. Some experiments on verb classification will be carried out in the $W_1L{\times}W_2$ space in Section 6.3.

Because the task of clustering concepts/words into superordinates has recently attracted much attention, we have three relevant data sets from the literature available for our tests. The Almuhareb–Poesio (AP) set includes 402 concepts from WordNet, balanced in terms of frequency and ambiguity. The concepts must be clustered into 21 classes, each selected from one of the 21 unique WordNet beginners, and represented by between 13 and 21 nouns. Examples include the *vehicle* class (*helicopter*, *motorcycle*...), the *motivation* class (*ethics*, *incitement*, ...), and the *social unit* class (*platoon*, *branch*). See Almuhareb (2006) for the full set.

The Battig test set introduced by Baroni et al. (2010) is based on the expanded Battig and Montague norms of Van Overschelde, Rawson, and Dunlosky (2004). The set comprises 83 concepts from 10 common concrete categories (up to 10 concepts per class), with the concepts selected so that they are rated as highly prototypical of the class. Class examples include *land mammals* (*dog*, *elephant*...), *tools* (*screwdriver*, *hammer*) and *fruit* (*orange*, *plum*). See Baroni et al. (2010) for the full list.

Finally, the ESSLLI 2008 set was used for one of the Distributional Semantic Workshop shared tasks (Baroni, Evert, and Lenci 2008). It is also based on concrete nouns, but it includes fewer prototypical members of categories (*rocket* as *vehicle* or *snail* as *land animal*). The 44 target concepts are organized into a hierarchy of classes of increasing abstraction. There are 6 lower level classes, with maximally 13 concepts per class (*birds*, *land animals*, *fruit*, *greens*, *tools*, *vehicles*). At a middle level, concepts are grouped into three classes (*animals*, *vegetables*, and *artifacts*). At the most abstract level, there is a two-way distinction between *living beings* and *objects*. See `http://wordspace.collocations.de` for the full set.

We cluster the nouns in each set by computing their similarity matrix based on pairwise cosines, and feeding it to the widely used CLUTO toolkit (Karypis 2003). We use CLUTO's built-in repeated bisections with global optimization method, accepting all of CLUTO's default values for this method.

Cluster quality is evaluated by percentage **purity**, one of the standard clustering quality measures returned by CLUTO (Zhao and Karypis 2003). If n_r^i is the number of items from the i-th true (gold standard) class that were assigned to the r-th cluster, n the total number of items, and k the number of clusters, then

$$\text{Purity} = \frac{1}{n}\sum_{r=1}^{k}\max_i(n_r^i)$$

Expressed in words, for each cluster we count the number of items that belong to the true class that is most represented in the cluster, and then we sum these counts across clusters. The resulting sum is divided by the total number of items so that, in the best case (perfect clusters), purity will be 1 (in percentage terms, 100%). As cluster quality deteriorates, purity approaches 0. For the models where we have full access to the results, we use a heuristic bootstrap procedure to obtain confidence intervals around the purities (Efron and Tibshirani 1994). We resample with replacement 10K data sets (cluster-assignment+true-label pairs) of the original size. Empirical 95% confidence intervals are then computed from the distribution of the purities in the bootstrapped data sets (for the ESSLLI results, we only perform the procedure for 6-way clustering). The confidence intervals give a rough idea of how stable purity estimates are across small variations of

the items in the data sets. The Random models for this task are baselines assigning the concepts randomly to the target clusters, with the constraint that each cluster must contain at least one concept. Random assignment is repeated 10K times, and we obtain means and confidence intervals from the distribution of these simulations.

Table 6 reports purity results for the three data sets, comparing our models to those in the literature. Again, the TypeDM model has an excellent performance. On the ESSLLI 2008 set, it outperforms the best configuration of the best shared task system among those that did three-level categorization (Katrenko's), despite the fact that the latter uses the full Web as a corpus and manually crafted patterns to improve feature extraction. TypeDM's performance is equally impressive on the AP set, where it outperforms AttrValue-05, the best unsupervised model by the data set proponents, trained on the full Web. Interestingly, the DepPath model of Rothenhäusler and Schütze (2009), which is the only one outperforming TypeDM on the AP set, is another structured model with dependency-based link-mediated features, which would fit well into the

Table 6
Purity in noun clustering with bootstrapped 95% confidence intervals (CI).

Almuhareb & Poesio (AP)					
model	*purity*	*95% CI*	*model*	*purity*	*95% CI*
DepPath[1]	79	NA	DV	65	61–69
TypeDM	76	72–81	DepDM	62	59–67
AttrValue-05[2]	71	NA	LexDM	59	56–65
Win	71	67–76	Random	16	14–17
VSM[3]	70	67–75			

Battig					
model	*purity*	*95% CI*	*model*	*purity*	*95% CI*
Win	96	91–100	DV-10[4]	79	73–89
TypeDM	94	89–99	LexDM	78	72–88
Strudel[4]	91	85–98	SVD-10[4]	71	67–83
DepDM	90	84–96	AttrValue[4]	45	44–61
DV	84	79–93	Random	29	24–34

ESSLLI 2008					
model	*6-way purity*	*95% CI*	*3-way purity*	*2-way purity*	*avg purity*
TypeDM	84	77–95	98	100	94.0
Katrenko[5]	91	NA	100	80	90.3
DV	75	70–89	93	100	89.3
DepDM	75	68–89	93	100	89.3
LexDM	75	70–89	87	100	87.3
Peirsman[5]	82	NA	84	86	84.0
Win	75	70–89	86	59	73.3
Shaoul[5]	41	NA	52	55	49.3
Random	38	32–45	49	57	48.0

Model sources: [1]Rothenhäusler and Schütze (2009); [2]Almuhareb and Poesio (2005); [3]Herdağdelen, Erk, and Baroni (2009); [4]Baroni et al. (2010); [5]ESSLLI 2008 shared task.

DM framework. TypeDM's purity is extremely high with the Battig set as well, although here it is outperformed by the unstructured Win model. Our top two performances are higher than Strudel, the best model by the proponents of the task. The latter was trained on about half of the data we used, however (moreover, the confidence intervals of these models largely overlap, suggesting that their difference is not significant).

6.1.4 Selectional Preferences. Our last pair of data sets for the $W_1{\times}LW_2$ space illustrate how the space can be used not only to measure similarity among words, but also to work with more abstract notions, such as that of a typical filler of an argument slot of a verb (such as the typical *killer* and the typical *killee*). We think that these are especially important experiments, because they show how the same matrix that has been used for tasks that were entirely bound to lexical items can also be used to generalize to structures that go beyond what is directly observed in the corpus. In particular, we model here selectional preferences (how plausible a noun is as subject/object of a verb), but our method is generalizable to many other semantic tasks that pertain to composition constraints; that is, they require measuring the goodness of fit of a word/concept as argument filler of another word/concept, including assigning semantic roles, logical metonymy, coercion (Pustejovsky 1995), and many other challenges.

The selectional preference test sets are based on averages of human judgments on a seven-point scale about the plausibility of nouns as arguments (either subjects or objects) of verbs. The McRae data set (McRae, Spivey-Knowlton, and Tanenhaus 1998) consists of 100 noun–verb pairs rated by 36 subjects. The Padó set (Padó 2007) has 211 pairs rated by 20 subjects.

For each verb, we first use the $W_1{\times}LW_2$ space to select a set of nouns that are highly associated with the verb via a subject or an object link. In this space, nouns are represented as vectors with dimensions that are labeled with $\langle link, word\rangle$ tuples, where the word might be a verb, and the link might stand for, among other things, syntactic relations such as *obj* (or, in the LexDM model, an expansion thereof, such as *obj+the-j*). To find nouns that are highly associated with a verb v when linked by the subject relation, we project the $W_1{\times}LW_2$ vectors onto a subspace where all dimensions are mapped to 0 except the dimensions that are labeled with $\langle l_{sbj}, v\rangle$, where l_{sbj} is a link containing either the string *sbj_intr* or the string *sbj_tr*, and v is the verb. We then measure the length of the noun vectors in this subspace, and pick the top n longest ones as prototypical subjects of the verb. The same operation is performed for the object relation. In our experiments, we set n to 20, but this is of course a parameter that should be explored.

We normalize and sum the vectors (in the full $W_1{\times}LW_2$ space) of the picked nouns, to obtain a centroid that represents an abstract "subject prototype" for the verb (and analogously for objects). The plausibility of an arbitrary noun as the subject (object) of a verb is then measured by the cosine of the noun vector to the subject (object) centroid in $W_1{\times}LW_2$ space. Crucially, the algorithm can provide plausibility scores for nouns that do not co-occur with the target verb in the corpus, by looking at how close they are to the centroid of nouns that do often co-occur with the verb. The corpus may contain neither *eat topinambur* nor *eat sympathy*, but the *topinambur* vector will likely be closer to the prototypical *eat* object vector than the one of *sympathy* would be.

It is worth stressing that the whole process relies on a single $W_1{\times}LW_2$ matrix: This space is first used to identify typical subjects (or objects) of a verb via subspacing, then to construct centroid vectors for the verb subject (object) prototypes, and finally to measure the distance of nouns to these centroids. Our method is essentially the same, save for implementation and parameter choice details, as the one proposed by Padó, Padó, and Erk (2007), in turn inspired by Erk (2007). However, they treat the identification

of typical argument fillers of a verb as an operation to be carried out using different resources, whereas we reinterpret it as a different way to use the same $W_1{\times}LW_2$ space in which we measure plausibility.

Following Padó and colleagues, we measure performance by the Spearman ρ correlation coefficient between the average human ratings and the model predictions, considering only verb–noun pairs that are present in the model. Table 7 reports percentage coverage and correlations for the DM models (the task requires the links to extract typical subjects and objects, so we cannot use DV nor Win), results from Padó, Padó, and Erk (2007) (ParCos is the best among their purely corpus-based systems), and the performance on the Padó data set of the supervised system of Herdağdelen and Baroni (2009). Testing for significance of the correlation coefficients with two-tailed tests based on a Spearman-coefficient derived t statistic, we find that the Resnik's model correlation for the McRae data is not significantly different from 0 ($t = 0.29$, $df = 92$, $p = 0.39$), ParCos on McRae is significant at $\alpha = .05$ ($t = 2.134$, $df = 89$, $p = 0.018$), and all other models on either data set are significant at $\alpha = .01$ and below.

TypeDM emerges as an excellent model to tackle selectional preferences, and as the overall winner on this task. On the Padó data set, it is as good as Padó's (2007) FrameNet based model, and it is outperformed only by the supervised BagPack approach. On the McRae data set, all three DM models do very well, and TypeDM is slightly worse than the other two models. On this data set, the DM models are outperformed by Padó's FrameNet model in terms of correlation, but the latter has a much lower coverage, suggesting that for practical purposes the DM models are a better choice. According to Raghunathan's test (see Section 6.1.1), the difference in correlation with human ratings among the three DM models is not significant on the McRae data, where TypeDM is below the other models ($Q = 0.19$, $df = 0.67$, $p = 0.50$). On the Padó data set, on the other hand, where TypeDM outperforms the other DM models, the same difference is highly significant ($Q = 12.70$, $df = 1.00$, $p < 0.001$).

As a final remark on the $W_1{\times}LW_2$ space, we can notice that DM models perform very well in tasks involving attributional similarity. The performance of unstructured DSMs (including Win, our own implementation of this type of model) is also high, sometimes even better than that of structured DSMs. However, our best DM model also achieves brilliant results in capturing selectional preferences, a task that is not directly addressable by unstructured DSMs. This fact suggests that the real advantage provided by structured DSMs—particularly when linguistic structure is suitably exploited, as

Table 7
Correlation with verb–argument plausibility judgments.

McRae			Padó		
model	*coverage*	ρ	*model*	*coverage*	ρ
Padó[1]	56	41	BagPack[2]	100	60
DepDM	97	32	TypeDM	100	51
LexDM	97	29	Padó[1]	97	51
TypeDM	97	28	ParCos[1]	98	48
ParCos[1]	91	22	DepDM	100	35
Resnik[1]	94	3	LexDM	100	34
			Resnik[1]	98	24

Model sources: [1]Padó, Padó, and Erk (2007); [2]Herdağdelen and Baroni (2009).

with the DM third-order tensor—actually resides in their versatility in addressing a much larger and various range of semantic tasks. This preliminary conclusion will also be confirmed by the experiments modeled with the other DM spaces.

6.2 The $W_1W_2{\times}L$ Space

The vectors of this space are labeled with word pair tuples $\langle w_1, w_2\rangle$ (columns of matrix $\mathbf{B}_{\text{mode-2}}$ in Table 3) and their dimensions are labeled with links l (rows of the same matrix). This arrangement of our tensor reproduces the "relational similarity" space of Turney (2006b), also implicitly assumed in much relation extraction work, where word pairs are compared based on the patterns that link them in the corpus, in order to measure the similarity of their relations (Pantel and Pennacchiotti 2006). The links that in $W_1{\times}LW_2$ space provide a form of shallow typing of lexical features ($\langle$*use, gun*$\rangle$) associated with single words (*soldier*) constitute under the $W_1W_2{\times}L$ view full features (*use*) associated with word pairs ($\langle$*soldier, gun*$\rangle$). Besides exploiting this view of the tensor to solve classic relational tasks, we will also show how problems that have not been traditionally defined in terms of a word-pair-by-link matrix, such as qualia harvesting with patterns or generating lists of characteristic properties, can be elegantly recast in the $W_1W_2{\times}L$ space by measuring the length of $\langle w_1, w_2\rangle$ vectors in a link (sub)space, thus bringing a wider range of semantic operations under the umbrella of the natural DM spaces.

The $W_1W_2{\times}L$ space represents pairs of words that co-occur in the corpus within the maximum span determined by the scope of the links connecting them (for our models, this maximum span is never larger than a single sentence). When words do not co-occur or only co-occur very rarely (and even in large corpora this will often be the case), attributional similarity can come to the rescue. Given a target pair, we can construct other, probably similar pairs by replacing one of the words with an attributional neighbor. For example, given the pair $\langle$*automobile, wheel*$\rangle$, we might discover in $W_1{\times}LW_2$ space that *car* is a close neighbor of *automobile*. We can then look for the pair $\langle$*car, wheel*$\rangle$, and use relational evidence about this pair as if it pertained to $\langle$*automobile, wheel*$\rangle$. This is essentially the way to deal with $W_1W_2{\times}L$ data sparseness proposed by Turney (2006b), except that he relies on independently harvested attributional and relational spaces, whereas we derive both from the same tensor. More precisely, in the $W_1W_2{\times}L$ tasks where we know the set of target pairs in advance (Sections 6.2.1 and 6.2.2), we smooth the DM models by combining in turn one of the words of each target pair with the top 20 nearest $W_1{\times}LW_2$ neighbors of the other word, obtaining a total of 41 pairs (including the original). The centroid of the $W_1W_2{\times}L$ vectors of these pairs is then taken to represent a target pair (the smoothed $\langle$*automobile, wheel*$\rangle$ vector is an average of the $\langle$*automobile, wheel*$\rangle$, $\langle$*car, wheel*$\rangle$, $\langle$*automobile, circle*$\rangle$, etc., vectors). The nearest neighbors are efficiently searched in the $W_1{\times}LW_2$ matrix by compressing it to 5,000 dimensions via random indexing, using the parameters suggested by Sahlgren (2005). Smoothing consistently improved performance, and we only report the relevant results for the smoothed versions of the models (including our implementation of LRA, to be discussed next).

We reimplemented Turney's Latent Relational Analysis (LRA) model, training it on our source corpus (LRA is trained separately for each test set, because it relies on a given list of word pairs to find the patterns that link them). We chose the parameter values of Turney's main model (his "baseline LRA system"). In short (see Turney's article for details), for a given set of target pairs we count all the patterns that connect them, in either order, in the corpus. Patterns are sequences of one to three words occurring between the targets, with all, none, or any subset of the elements replaced by wildcards (*with the, with*

Table 8
Percentage accuracy in solving SAT analogies with 95% binomial confidence intervals (CI).

model	*accuracy*	*95% CI*	*model*	*accuracy*	*95% CI*
Human[1]	57.0	52.0–62.3	TypeDM	42.4	37.4–47.7
LRA-06[2]	56.1	51.0–61.2	LSA[7]	42.0	37.2–47.4
PERT[3]	53.3	48.5–58.9	LRA	37.8	32.8–42.8
PairClass[4]	52.1	46.9–57.3	PMI-IR-06[2]	35.0	30.2–40.1
VSM[1]	47.1	42.2–52.5	DepDM	31.4	26.6–36.2
BagPack[5]	44.1	39.0–49.3	LexDM	29.3	24.8–34.3
k-means[6]	44.0	39.0–49.3	Random	20.0	16.1–24.5

Model sources: [1]Turney and Littman (2005); [2]Turney (2006b); [3]Turney (2006a); [4]Turney (2008); [5]Herdağdelen and Baroni (2009); [6]Biciçi and Yuret (2006); [7]Quesada, Mangalath, and Kintsch (2004).

*, * *the*, * *). Only the top 4,000 most frequent patterns are preserved, and a target-pair-by-pattern matrix is constructed (with 8,000 dimensions, to account for directionality). Values in the matrix are log- and entropy-transformed using Turney's formula. Finally, SVD is applied, reducing the columns to the top 300 latent dimensions (here and subsequently, we use SVDLIBC[9] to perform SVD). For simplicity and to make LRA more directly comparable to the DM models, we applied our attributional-neighbor-based smoothing technique (the neighbors for target pair expansion are taken from the best attributional DM model, namely, TypeDM) instead of the more sophisticated one used by Turney. Thus, our LRA implementation differs from Turney's original in two aspects: the smoothing method and the source corpus (Turney uses a corpus of more than 50 billion words). Neither variation pertains to inherent differences between LRA and DM. Given the appropriate resources, a DM model could be trained on Turney's gigantic corpus, and smoothed with his technique.

6.2.1 Solving Analogy Problems. The SAT test set introduced by Turney and collaborators contains 374 multiple-choice questions from the SAT college entrance exam. Each question includes one target (*ostrich–bird*) and five candidate analogies (*lion–cat*, *goose–flock*, *ewe–sheep*, *cub–bear*, *primate–monkey*). The data set is dominated by noun–noun pairs, but all other combinations are also attested (noun–verb, verb–adjective, verb–verb, etc.) The task is to choose the candidate pair most analogous to the target (*lion–cat* in the previous example). This is essentially the same task as the TOEFL, but applied to word pairs instead of words. As in the TOEFL, we pick the candidate with the highest cosine with the target as the right analogy.

Table 8 reports our SAT results together with those of other corpus-based methods from the ACL Wiki and other systems. TypeDM is again emerging as the best among our models. To put its performance in context statistically, according to a Fisher test its accuracy is not significantly different from that of VSM ($p = 0.239$), whereas it is better than that of PMI-IR-06 ($p = 0.043$; even the bottom model, LexDM, is significantly better than the random guesser, $p = 0.004$).

TypeDM is at least as good as LRA when the latter is trained on the same data and smoothed with our method, suggesting that the excellent performance of Turney's version of LRA (LRA-06) is due to the fact that he used a much larger corpus, and/or to

9 http://tedlab.mit.edu/∼dr/SVDLIBC/.

his more sophisticated smoothing technique, and not to the specific way in which LRA collects corpus-based statistics. All the algorithms with higher accuracy than TypeDM are based on much larger input corpora, except BagPack, which is, however, supervised. The LSA system of Quesada, Mangalath, and Kintsch (2004), which performs similarly to TypeDM, is based on a smaller corpus, but it relies on hand-coded "analogy domains" that are represented by lists of manually selected characteristic words.

6.2.2 Relation Classification. Just as the SAT is the relational equivalent of the TOEFL task, the test sets we tackle next are a relational analog to attributional concept clustering, in that they require grouping pairs of words into classes that instantiate the same relations. Whereas we cast attributional categorization as an unsupervised clustering problem (following much of the earlier literature), the common approach to classifying word pairs by relation is supervised, and relies on labeled examples for training. In this article, we exploit training data in a very simple way, via a *nearest centroid method*. In the SEMEVAL task we are about to introduce, where both positive and negative examples are available for each class, we use the positive examples to construct a centroid that represents a target class, and negative examples to construct a centroid representing items outside the class. We then decide if a test pair belongs to the target class by measuring its distance from the positive and negative centroids, picking the nearest one. For example, the Cause–Effect relation has positive training examples such as *cycling–happiness* and *massage–relief* and negative examples such as *customer–satisfaction* and *exposure–protection*. We create a positive centroid by summing the $W_1W_2{\times}L$ vectors of the first set of pairs, and a negative centroid by summing the latter. We then measure the cosine of a test item such as *smile–wrinkle* with the centroids, and decide if it instantiates the Cause–Effect relation based on whether it is closer to the positive or negative centroid. For the other tasks (as well as the transitive alternation task of Section 6.3), we do not have negative examples, but positive examples for different classes. We create a centroid for each class, and classify test items based on the centroid they are nearest to.

Our first test pertains to the seven relations between nominals in Task 4 of SEMEVAL 2007 (Girju et al. 2007): Cause–Effect, Instrument–Agency, Product–Producer, Origin–Entity, Theme–Tool, Part–Whole, Content–Container. For each relation, the data set includes 140 training and about 80 test items. Each item consists of a Web snippet, containing word pairs connected by a certain pattern (e.g., "* *causes* *"). The retrieved snippets are manually classified by the SEMEVAL organizers as positive or negative instances of a certain relation (see the earlier Cause–Effect examples). About 50% training and test cases are positive instances. In our experiments we do not make use of the contexts of the target word pairs that are provided with the test set.

The second data set (NS) comes from Nastase and Szpakowicz (2003). It pertains to the classification of 600 modifier–noun pairs and it is of interest because it proposes a very fine-grained categorization into 30 semantic classes, such as Cause (*cloud–storm*), Purpose (*album–picture*), Location-At (*pain–chest*), Location-From (*visitor–country*), Frequency (*superstition–occasional*), Time-At (*snack–midnight*), and so on. The modifiers can be nouns, adjectives, or adverbs. Because the data set is not split into training and test data we follow Turney (2006b) and perform leave-one-out cross-validation. The data set also comes with a coarser five-way classification. Our unreported results on it are comparable, in terms of relative performance, to the ones for the 30-way classification.

The last data set (OC) contains 1,443 noun–noun compounds classified by Ó Séaghdha and Copestake (2009) into 6 relations: Be (*celebrity–winner*), Have (*door–latch*), In (*air–disaster*), Actor (*university–scholarship*), Instrument (*freight–train*), and About

(*bank–panic*); see Ó Séaghdha and Copestake (2009) and references there. We use the same five-way cross-validation splits as the data set proponents.

Table 9 reports performance of models from our experiments and from the literature on the three supervised relation classification tasks. Following the relevant earlier studies, for SEMEVAL we report macro-averaged accuracy, whereas for the other two data sets we report global accuracy (with binomial confidence intervals). All other measures are macro-averaged. Majority is the performance of a classifier that always guesses the

Table 9
Relation classification performance; all measures macro-averaged, except accuracy in the NS and OC data sets, where we also report the accuracy 95% confidence intervals (CI).

SEMEVAL 2007

model	*prec*	*recall*	*F*	*acc*	*model*	*prec*	*recall*	*F*	*acc*
TypeDM	71.7	62.5	66.4	70.2	DepDM	61.0	57.3	58.9	61.8
UCD-FC[1]	66.1	66.7	64.8	66.0	UTH[1]	56.1	57.1	55.9	58.8
UCB[1]	62.7	63.0	62.7	65.4	Majority	81.3	42.9	30.8	57.0
LexDM	64.7	61.3	62.5	65.4	ProbMatch	48.5	48.5	48.5	51.7
ILK[1]	60.5	69.5	63.8	63.5	UC3M[1]	48.2	40.3	43.1	49.9
LRA	63.7	60.0	61.0	63.1	AllTrue	48.5	100.0	64.8	48.5
UMELB-B[1]	61.5	55.7	57.8	62.7					

Nastase & Szpakowicz (NS)

model	*prec*	*recall*	*F*	*acc*	*acc 95% CI*
LRA-06[2]	41.0	35.9	36.6	39.8	35.9–43.9
VSM-AV[3]	27.9	26.8	26.5	27.8	24.3–31.6
LRA	23.0	23.1	21.1	25.5	22.1–29.2
VSM-WMTS[2]	24.0	20.9	20.3	24.7	21.3–28.3
TypeDM	19.5	20.2	13.7	15.4	12.5–18-5
LexDM	7.5	14.1	8.1	12.1	9.7–15.0
DepDM	11.6	14.5	8.1	8.7	6.5–11.2
Majority	0.3	3.3	0.5	8.2	6.1–10.6
ProbMatch	3.3	3.3	3.3	4.7	3.1–6.7
AllTrue	3.3	100	6.4	NA	NA

Ó Séaghdha & Copestake (OC)

model	*prec*	*recall*	*F*	*acc*	*acc 95% CI*
OC-Comb[4]	NA	NA	61.6	63.1	60.6–65.6
OC-Rel[4]	NA	NA	49.9	52.1	49.5–54.7
TypeDM	33.8	33.5	31.4	32.1	29.7–34.6
LRA	31.5	30.8	30.7	31.3	28.9–33.8
LexDM	29.9	28.9	28.7	29.7	27.4–32.2
DepDM	28.2	28.2	27.0	27.6	25.3–30.0
Majority	3.6	16.7	5.9	21.3	19.2–23.5
ProbMatch	16.7	16.7	16.7	17.1	15.2–19.2
AllTrue	16.7	100	28.5	NA	NA

Model sources: [1]SEMEVAL Task 4; [2]Turney (2006b); [3]Turney and Littman (2005); [4]Ó Séaghdha and Copestake (2009).

majority class in the test set (in SEMEVAL, for each class, it guesses that all or no items belong to it depending on whether there are more positive or negative examples in the test data; in the other tasks, it labels all items with the majority class). AllTrue always assigns an item to the target class (being inherently binary, it does not provide a well-defined multi-class global accuracy). ProbMatch randomly guesses classes matching their distribution in the test data (in SEMEVAL, it matches the proportion of positive and negative examples within each class).

For SEMEVAL, the table reports the results of those models that took part in the shared task and, like ours, did not use the organizer-provided WordNet sense labels nor information about the query used to retrieve the examples. All these models are outperformed by TypeDM, despite the fact that they exploit the training contexts and/or specific additional resources: an annotated compound database (UCD-FC), more sophisticated machine learning algorithms to train the relation classifiers (ILK, UCD-FC), Web counts (UCB), and so on.

For the NS data set, none of the DM models do well, although TypeDM is once more the best among them. The DM models are outperformed by other models from the literature, all trained on much larger corpora, and also by our implementation of LRA. The difference in global accuracy between LRA and TypeDM is significant (Fisher test, $p = 0.00002$). TypeDM's accuracy is nevertheless well above the best (Majority) baseline accuracy ($p = 0.0001$).

The OC results confirm that TypeDM is the best of our models, again (slightly) outperforming our LRA implementation. Still, our best performance is well below that of OC-Comb, the absolute best, and OC-Rel, the best purely relational model of Ó Séaghdha and Copestake (2009) (the difference in global accuracy between the latter and TypeDM is highly significant, $p < 0.000001$). Ó Séaghdha and Copestake use sophisticated kernel-based methods and extensive parameter tuning to achieve these results. We hope that the TypeDM performance would also improve by improving the machine learning aspects of the procedure.

As an ad interim summary, we observe that TypeDM achieves competitive results in semantic tasks involving relational similarity. In particular, in both analogy solving and two out of three relation classification experiments, TypeDM is at least as good as our LRA implementation. We now move on to show how this same view of the DM tensor can be successfully applied to aspects of meaning that are not normally addressed by relational DSMs.

6.2.3 Qualia Extraction. A popular alternative to the supervised approach to relation extraction is to pick a set of lexico-syntactic patterns that should capture the relation of interest and to harvest pairs they connect in text, as famously illustrated by Hearst (1992) for the hyponymy relation. In the DM approach, instead of going back to the corpus to harvest the patterns, we exploit the information already available in the $W_1W_2{\times}L$ space. We select promising links as our equivalent of patterns and we measure the length of word pair vectors in the $W_1W_2{\times}L$ subspace defined by these links. We illustrate this with the data set of Cimiano and Wenderoth (2007), which contains **qualia structures** (Pustejovsky 1995) for 30 nominal concepts, both concrete (*door*) and abstract (*imagination*). Cimiano and Wenderoth asked 30 subjects to produce qualia for these words (each word was rated by at least three subjects), obtaining a total of 1,487 word–quale pairs, instantiating the four roles postulated by Pustejovsky: Formal (the category of the object: *door–barrier*), Constitutive (constitutive parts, materials the object is made of: *food–fat*), Agentive (what brings the object about: *letter–write*), and Telic (the function of the object: *novel–entertain*).

We approximate the patterns proposed by Cimiano and Wenderoth by manually selecting links that are already in our DM models, as reported in Table 10 (here and subsequently when discussing qualia-harvesting links, we use *n* and *q* to indicate the linear position of the noun and the potential quale with respect to the link). All qualia roles have links pertaining to noun–noun pairs. The Agentive and Telic patterns also harvest noun–verb pairs. For LexDM, we pick all links that begin with one of the strings in Table 10. For the DepDM model, the only attested links are *n with q* (Constitutive), *n sbj_intr q*, *n sbj_tr q* (Telic), and *q obj n* (Agentive). Consequently, DepDM does not harvest Formal qualia, and is penalized accordingly in the evaluation.

We project all $W_1W_2{\times}L$ vectors that contain a target noun onto each of the four subspaces determined by the quale-specific link sets, and we compute their subspace lengths. Given a target noun *n* and a potential quale *q*, the length of the $\langle n, q\rangle$ vector in the subspace characterized by the links that represent role *r* is our measure of how good *q* is as a quale of type *r* for *n* (for example, the length of $\langle book, read\rangle$ in the subspace defined by the Telic links is our measure of fitness of *read* as Telic role of *book*). We use length in the subspace associated to the qualia role *r* to rank all $\langle n, q\rangle$ pairs relevant to *r*.

Following Cimiano and Wenderoth's evaluation method, for each noun we first compute, separately for each role, the ranked list precision (with respect to the manually constructed qualia structure) at 11 equally spaced recall levels from 0% to 100%. We select the precision, recall, and F values at the recall level that results in the highest F score (i.e., in the best precision–recall trade-off). We then average across the roles, and then across target nouns. The task, as framed here, cannot be run with the LRA model, and, because of its open-ended nature (we do not start from a predefined list of pairs), we do not smooth the models.

Table 11 reports the performance of our models, as well as the F scores reported by Cimiano and Wenderoth. For our models, where we have access to the itemized data, we also report the standard deviation of F across the target nouns.

All the DM models perform well (including DepDM, which is disfavored by the lack of Formal links), and once more TypeDM emerges as the best among them, with an F value that is also (slightly) above the best Cimiano and Wenderoth models (that are based on co-occurrence counts from the whole Web). Despite the large standard deviations, the difference in F across concepts between TypeDM and the second-best DM model (DepDM) is highly significant (paired t-test, $t = 4.02$, $df = 29$, $p < 0.001$), suggesting that the large variance is due to different degrees of difficulty of the concepts, affecting the models in similar ways.

Table 10
Links approximating the patterns proposed in Cimiano and Wenderoth (2007).

FORMAL	*CONSTITUTIVE*
n as-form-of *q*, *q* as-form-of *n* *n* as-kind-of *q*, *n* as-sort-of *q*, *n* be *q* *q* such_as *n*	*q* as-member-of *n*, *q* as-part-of *n*, *n* with *q* *n* with-lot-of *q*, *n* with-majority-of *q* *n* with-number-of *q*, *n* with-sort-of *q* *n* with-variety-of *q*
AGENTIVE	*TELIC*
n as-result-of *q*, *q* obj *n*	*n* for-use-as *q*, *n* for-use-in *q*, *n* sbj_tr *q* *n* sbj_intr *q*

Table 11
Average qualia extraction performance.

model	*precision*	*recall*	*F*	*F s.d.*
TypeDM	26.2	22.7	18.4	8.7
P[1]	NA	NA	17.1	NA
WebP[1]	NA	NA	16.7	NA
LexDM	19.9	23.6	16.2	7.1
WebJac[1]	NA	NA	15.2	NA
DepDM	17.8	16.9	12.8	6.4
Verb-PMI[1]	NA	NA	10.7	NA
Base[1]	NA	NA	7.6	NA

Model source: [1]Cimiano and Wenderoth (2007).

6.2.4 Predicting Characteristic Properties. Recently, there has been some interest in the automated generation of commonsense concept descriptions in terms of intuitively salient properties: a *dog* is a *mammal*, it *barks*, it has a *tail*, and so forth (Almuhareb 2006; Baroni and Lenci 2008; Baroni, Evert, and Lenci 2008; Baroni et al. 2010). Similar property lists, collected from subjects in elicitation tasks, are widely used in cognitive science as surrogates of mental features (Garrard et al. 2001; McRae et al. 2005; Vinson and Vigliocco 2008). Large-scale collections of property-based concept descriptions are also carried out in AI, where they are important for commonsense reasoning (Liu and Singh 2004).

In the qualia task, given a concept we had to extract properties of certain kinds (corresponding to the qualia roles). The property-based description task is less constrained, because the most salient relations of a nominal concept might be in all sorts of relations with it (parts, typical behaviors, location, etc.). Still, we couch the task of unconstrained property extraction as a challenge in the $W_1W_2{\times}L$ space. The approach is similar to the method adopted for qualia roles, but now the whole $W_1W_2{\times}L$ space is used, instead of selected subspaces. Given all the $\langle n, w_2\rangle$ pairs that have the target nominal concept as first element, we rank them by length in the $W_1W_2{\times}L$ space. The longest $\langle n, w_2\rangle$ vectors in this space should correspond to salient properties of the target concept, as we expect a concept to often co-occur in texts with its important properties (because in the current DM implementations links are disjoint across POS, we map properties with different POS onto the same scale by dividing the length of the vector representing a pair by the length of the longest vector in the harvested concept–property set that has the same POS pair). For example, among the longest $W_1W_2{\times}L$ vectors with *car* as first item we find $\langle$*car, drive*$\rangle$, $\langle$*car, park*$\rangle$, and $\langle$*car, engine*$\rangle$. The first two pairs are normalized by dividing by the longest $\langle$*noun, verb*$\rangle$ vector in the harvested set, the third by dividing by the longest $\langle$*noun, noun*$\rangle$ vector.

We test this approach in the ESSLLI 2008 Distributional Semantic Workshop unconstrained property generation challenge (Baroni, Evert, and Lenci 2008). The data set contains, for each of 44 concrete concepts, 10 properties that are those that were most frequently produced by subjects in the elicitation experiment of McRae et al. (2005) (the "gold standard lists"). Algorithms must generate lists of 10 properties per concept, and performance is measured by overlap with the subject-produced properties, that is, by the cross-concept average proportions of properties in the generated lists that are also in the corresponding gold standard lists. Smoothing would be very costly (we would need to smooth all pairs that contain a target concept) and probably counterproductive

(as the most typical properties of a concept should be highly specific to it, rather than shared with neighbors). Because LRA (at least in a reasonably efficient implementation) requires a priori specification of the target pairs, it is not well suited to this task.

Table 12 reports the percentage overlap with the gold standard properties (averaged across the 44 concepts) for our models as well as the only ESSLLI 2008 participant that tried this task, and for the models of Baroni et al. (2010). TypeDM is the best DM model, and it also does quite well compared to the state of the art. The difference between Strudel, the best model from the earlier literature, and TypeDM is not statistically significant, according to a paired t-test across the target concepts ($t = 1.1$, $df = 43$, $p = 0.27$). The difference between TypeDM and DV-10, the second best model from the literature, is highly significant ($t = 2.9$, $df = 43$, $p < 0.01$). If we consider how difficult this sort of open-ended task is (see the very low performance of the respectable models at the bottom of the list), matching on average two out of ten speaker-generated properties, as TypeDM does, is an impressive feat.

6.3 The $W_1L\times W_2$ Space

The vectors of this space are labeled with binary tuples of type $\langle w_1, l\rangle$ (columns of matrix $\mathbf{C}_{\text{mode-3}}$ in Table 3), and their dimensions are labeled with words w_2 (rows of the same matrix). We illustrate this space in the task of discriminating verbs participating in different argument alternations. However, other uses of the space can also be foreseen. For example, the rows of $W_1L\times W_2$ correspond to the columns of the $W_1\times LW_2$ space (given the constraints on the tuple structure we adopted in Section 3.1). We could use the former space for feature smoothing or selection in the latter space, for example, by merging the features of $W_1\times LW_2$ whose corresponding vectors in $W_1L\times W_2$ have a cosine similarity over a given threshold. We leave this possibility to further work.

Among the linguistic objects represented by the $W_1L\times W_2$ vectors, we find the syntactic slots of verb frames. For instance, the vector labeled with the tuple $\langle$*read*, $sbj^{-1}\rangle$ represents the subject slot of the verb *read* in terms of the distribution of its noun fillers, which label the dimensions of the space. We can use the $W_1L\times W_2$ space to explore the semantic properties of syntactic frames, and to extract generalizations about the inner structure of lexico-semantic representations of the sort formal semanticists have traditionally been interested in. For instance, the high similarity between the object slot of *kill* and the subject slot of *die* might provide a distributional correlate to the classic *cause(subj,die(obj))* analysis of *killing* by Dowty (1977) and many others.

Measuring the cosine between the vectors of different syntactic slots of the same verb corresponds to estimating the amount of fillers they share. Measures of "slot overlap" have been used by Joanis, Stevenson, and James (2008) as features to classify verbs on the basis of their argument alternations. Levin and Rappaport-Hovav (2005)

Table 12
Average percentage overlap with subject-generated properties and standard deviation.

model	*overlap*	*s.d.*	*model*	*overlap*	*s.d.*	*model*	*overlap*	*s.d.*
Strudel[1]	23.9	11.3	LexDM	14.5	12.1	SVD-10[1]	4.1	6.1
TypeDM	19.5	12.4	DV-10[1]	14.1	10.3	Shaoul[2]	1.8	3.9
DepDM	16.1	12.6	AttrValue[1]	8.8	9.9			

Model sources: [1]Baroni et al. (2010); [2]ESSLLI 2008 shared task.

define **argument alternations** as the possibility for verbs to have multiple syntactic realizations of their semantic argument structure. Alternations involve the expression of the same semantic argument in two different syntactic slots. We expect that, if a verb undergoes a particular alternation, then the set of nouns that appear in the two alternating slots should overlap to a certain degree.

Argument alternations represent a key aspect of the complex constraints that shape the syntax–semantics interface. Verbs differ with respect to the possible alternations they can undergo, and this variation is strongly dependent on their semantic properties (semantic roles, event type, etc.). Levin (1993) has in fact proposed a well-known classification of verbs based on their range of syntactic alternations. Recognizing the alternations licensed by a verb is extremely important in capturing its argument structure properties, and consequently in describing its semantic behavior. We focus here on a particular class of alternations, namely transitivity alternations, whose verbs allow both for a transitive NP V NP variant and for an intransitive NP V (PP) variant (Levin 1993). We use the $W_1L{\times}W_2$ space to carry out the automatic classification of verbs that participate in different types of transitivity alternations.

In the *causative/inchoative* alternation, the object argument (e.g., *John broke the vase*) can also be realized as an intransitive subject (e.g., *The vase broke*). In a first experiment, we use the $W_1L{\times}W_2$ space to discriminate between transitive verbs undergoing the causative/inchoative alternation (C/I) (e.g., *break*) and non-alternating ones (e.g., *mince*; cf. *John minced the meat* vs. **The meat minced*). The C/I data set was introduced by Baroni and Lenci (2009), but not tested in a classification task there. It consists of 232 causative/inchoative verbs and 170 non-alternating transitive verbs from Levin (1993).

In a second experiment, we apply the $W_1L{\times}W_2$ space to discriminate verbs that belong to three different classes, each corresponding to a different type of transitive alternation. We use the MS data set (Merlo and Stevenson 2001), which includes 19 unergative verbs undergoing the induced action alternation (e.g., *race*), 19 unaccusative verbs that undergo the causative/inchoative alternation (e.g., *break*), and 20 object-drop verbs participating in the unexpressed object alternation (e.g., *play*). See Levin (1993) for details about each of these transitive alternations. The complexity of this task is due to the fact that the verbs in the three classes have both transitive and intransitive variants, but with very different semantic roles. For instance, the transitive subject of unaccusative (*The man broke the vase*) and unergative verbs (*The jockey raced the horse past the barn*) is an agent of causation, whereas the subject of the intransitive variant of unaccusative verbs has a theme role (i.e., undergoes a change of state: *The vase broke*), and the intransitive subject of unergative verbs has instead an agent role (*The horse raced past the barn*). Thus, their surface identity notwithstanding, the semantic properties of the syntactic slots of the verbs in each class are very different. By testing the $W_1L{\times}W_2$ space on such a task we can therefore evaluate its ability to capture non-trivial properties of the verb's thematic structure.

We address these tasks by measuring the similarities between the $W_1L{\times}W_2$ vectors of the transitive subject, intransitive subject, and direct object slots of a verb, and using these inter-slot similarities to classify the verb. For instance, given the definition of the C/I alternation, we can predict that with alternating verbs the intransitive subject slot should be similar to the direct object slot (the things that are broken also break), while this should not hold for non-alternating verbs (*mincees* are very different from *mincers*). For each verb v in a data set, we extract the corresponding $W_1L{\times}W_2$ slot vectors $\langle v,l\rangle$ whose links are *sbj_intr*, *sbj_tr*, and *obj* (for LexDM, we sum the vectors with links beginning with one of these three patterns). Then, for each v we build a three-dimensional vector with the cosines between the three slot vectors. These second

order vectors encode the profile of similarity across the slots of a verb, and can be used to spot verbs that have comparable profiles (e.g., verbs that have a high similarity between their *subj_intr* and *obj* slots).

We model both experiments as classification tasks using the nearest centroid method on the three-dimensional vectors, with leave-one-out cross-validation. We perform binary classification of the C/I data set (treating non-alternating verbs as negative examples), and three-way classification of the MS data. Table 13 reports the results, with the baselines computed similarly to the ones in Section 6.2.2 (for C/I, Majority is equivalent to AllTrue). The DM performance is also compared with the results of Merlo and Stevenson (2001) for their classifiers tested with the leave-one-out methodology (macro-averaged F has been computed on the class-by-class scores reported in that article).

All the DM models discriminate the verb classes much more reliably than the baselines. The accuracy of DepDM, the worst DM model, is significantly higher than that of the best baselines, AllTrue in C/I (Fisher test, $p = 0.024$) and Majority on MS ($p = 0.039$).

TypeDM is again our best model. Its performance is comparable to the lower range of the Merlo and Stevenson classifiers (considering the large confidence intervals due to the small sample size, the accuracy of TypeDM is not significantly below even that of the top model NoPass; $p = 0.43$). The TypeDM results were obtained simply by measuring the verb inter-slot similarities in the $W_1L{\times}W_2$ space. Conversely, the classifiers in Merlo and Stevenson (2001) rely on a much larger range of knowledge-intensive features selected in an ad hoc fashion for this task (on the other hand, their training corpus

Table 13
Verb classification performance (precision, recall, and F for MS are macro-averaged). Global accuracy supplemented by 95% binomial confidence intervals (CI).

	Causative/Inchoative (C/I)				
model	*prec*	*recall*	*F*	*acc*	*acc 95% CI*
LexDM	76.0	69.9	72.8	69.9	65.2–74.3
TypeDM	75.7	68.5	71.9	69.1	64.4–73.6
DepDM	72.8	64.6	68.4	65.7	60.8–70.3
AllTrue	57.7	100	73.2	57.7	52.7–62.6
ProbMatch	57.7	57.7	57.7	51.2	46.2–56.2

	Merlo & Stevenson (MS)				
model	*prec*	*recall*	*F*	*acc*	*acc 95% CI*
NoPass[1]	NA	NA	71.2	71.2	57.3–81.9
AllFeatures[1]	NA	NA	69.1	69.5	55.5–80.5
NoTrans[1]	NA	NA	63.8	64.4	50.1–76.0
NoCaus[1]	NA	NA	62.6	62.7	48.4–74.5
TypeDM	60.7	61.7	60.8	61.5	47.5–73.7
NoVBN[1]	NA	NA	61.0	61.0	46.6–73.0
NoAnim[1]	NA	NA	59.9	61.0	46.6–73.0
LexDM	55.3	56.7	55.8	56.4	43.2–69.8
DepDM	52.9	55.0	53.2	54.7	41.5–68.3
Majority	11.3	33.3	16.9	33.9	22.5–48.1
ProbMatch	33.3	33.3	33.3	33.3	21.0–46.3
AllTrue	33.3	100	50.0	NA	NA

Model source: [1]Merlo and Stevenson (2001).

is not parsed and it is much smaller than ours). Finally, we can notice that in both experiments the mildly (TypeDM) and heavily (LexDM) lexicalized DM models score better than their non-lexicalized counterpart (DepDM), although the difference between the best DM model and DepDM is not significant on either data set ($p = 0.23$ for the LexDM/DepDM difference in C/I; $p = 0.57$ for the TypeDM/DepDM difference in MS).

Verb alternations do not typically appear among the standard tasks on which DSMs are tested. Moreover, they involve non-trivial properties of argument structure. The good performance of DM in these experiments is therefore particularly significant in supporting its vocation as a general model for distributional semantics.

6.4 The $L{\times}W_1W_2$ Space

The vectors of this space are labeled with links l (rows of matrix $\mathbf{B}_{\text{mode-2}}$ in Table 3) and their dimensions are labeled with word pair tuples $\langle w_1, w_2 \rangle$ (columns of the same matrix). Links are represented in terms of the word pairs they connect. The $L{\times}W_1W_2$ space supports tasks where we are directly interested in the links as an object of study—for example, characterizing prepositions (Baldwin, Kordoni, and Villavicencio 2009) or measuring the relative similarity of different kinds of verb–noun relations. We focus here instead on a potentially more common use of $L{\times}W_1W_2$ vectors as a "feature selection and labeling" space for $W_1W_2{\times}L$ tasks.

Specifically, we go back to the qualia extraction task of Section 6.2.3. There, we started with manually identified links. Here, we start with examples of noun–quale pairs $\langle n, q_r \rangle$ that instantiate a role r. We project all $L{\times}W_1W_2$ vectors in a subspace where only dimensions corresponding to one of the example pairs are non-zero. We then pick the most characteristic links in this subspace to represent the target role r, and look for new pairs $\langle n, q_r \rangle$ in the $W_1W_2{\times}L$ subspace defined by these automatically picked links, instead of the manual ones. Although we stop at this point, the procedure can be seen as a DM version of popular iterative bootstrapping algorithms such as Espresso (Pantel and Pennacchiotti 2006): Start with some examples of the target relation, find links that are typical of these examples, use the links to find new examples, and so on. In DM, the process does not go back to a corpus to harvest new links and example pairs, but it iterates between the column and row spaces of a pre-compiled matrix (i.e, the mode-2 matricization in Table 3).

For each of the 30 noun concepts in the Cimiano and Wenderoth gold standard, we use the noun–quale pairs pertaining to the remaining 29 concepts as training examples to select a set of 20 links that we then use in the same way as the manually selected links of Section 6.2.3. Simply picking the longest links in the $L{\times}W_1W_2$ subspace defined by the example $\langle n, q_r \rangle$ dimensions does not work, because we harvest links that are frequent in general, rather than characteristic of the qualia roles (noun modification, *of*, etc.). For each role r, we construct instead two $L{\times}W_1W_2$ subspaces, one positive subspace with the example pairs $\langle n, q_r \rangle$ as unique non-zero dimensions, and a negative subspace with non-zero dimensions corresponding to all $\langle w_1, w_2 \rangle$ pairs such that w_1 is one of the training nominal concepts, and w_2 is not a quale q_r in the example pairs. We then measure the length of each link in both subspaces. For example, we measure the length of the *obj* link in a subspace characterized by $\langle n, q_{telic} \rangle$ example pairs, and the length of *obj* in a subspace characterized by $\langle n, w_2 \rangle$ pairs that are probably *not* Telic examples. We compute the pointwise mutual information (PMI) statistic (Church and Hanks 1990) on these lengths to find the links that are most typical of the positive subspace corresponding to each qualia role. PMI, with respect to other association measures, finds more specific links, which is good for our purposes. However, it is also notoriously

prone to over-estimating the importance of rare items (Manning and Schütze 1999, Chapter 5). Thus, before selecting the top 20 links ranked by PMI, we filter out those links that do not have at least 10 non-zero dimensions in the positive subspace. Many parameters here should be tuned more systematically (top n links, association measure, minimum non-zero dimensions), but the current results will nevertheless illustrate our methodology.

Table 14 reports, for each quale, the TypeDM links that were selected in each of the 30 leave-one-concept-out folds. The links *n is q*, *n in q*, and *q such_as n* are a good sketch of the Formal relation, which essentially subsumes various taxonomic relations. The other Formal links are less conspicuous. However, note the presence of noun coordination (*n coord q* and *q coord n*), consistently with the common claim that coordinated terms tend to be related taxonomically (Widdows and Dorow 2002). Constitutive is mostly a whole–part relation, and the harvested links do a good job at illustrating such a relation. For the Telic, *q by n*, *q through n*, and *q via n* capture cases in which the quale stands in an action–instrument relation to the target noun (*murder by knife*). These links thus encode the subtype of Telic role that Pustejovsky (1995) calls "indirect." The two verb–noun links (*q obj n* and *n sbj_intr q*) instead capture "direct" Telic roles, which are typically expressed by the theme of a verb (*read a book*, *the book reads well*). The least convincing results are those for the Agentive role, where only *q obj n* and perhaps *q out n* are intuitively plausible canonical links. Interestingly, the manual selections we carried out in Section 6.2.3 also gave very poor results for the Agentive role, as shown by the fact that Table 10 reports just one link for such a role. This suggests that the problems with this qualia role might be due to the number and type of lexicalized links used to build the DM tensors, rather than to the selection algorithm presented here.

Coming now to the quantitative evaluation of the harvested patterns, the results in Table 15 (to be compared to Table 11 in Section 6.2.3) are based on $W_1W_2{\times}L$ subspaces where the non-zero dimensions correspond to the links that we picked automatically with the method we just described (different links for each concept, because of the leave-one-concept-out procedure). TypeDM is the best model in this setting as well. Its performance is even better than the one (reported in Table 11) obtained with the manually picked patterns (although the difference is not statistically significant; paired t-test, $t = 0.75$, $df = 29$, $p = 0.46$), and the automated approach has more room for improvement via parameter optimization.

We did not get as deeply into $L{\times}W_1W_2$ space as we did with the other views, but our preliminary results on qualia harvesting suggest at least that looking at links as

Table 14
Links selected in all folds of the leave-one-out procedure to extract links typical of each qualia role.

FORMAL	*CONSTITUTIVE*
n is *q*, *q* is *n*, *q* become *n*, *n* coord *q*, *q* coord *n*, *q* have *n*, *n* in *q*, *n* provide *q*, *q* such_as *n*	*n* have *q*, *n* use *q*, *n* with *q*, *n* without *q*
AGENTIVE	*TELIC*
q after *n*, *q* alongside *n*, *q* as *n*, *q* before *n*, *q* besides *n*, *q* during *n*, *q* in *n*, *q* obj *n*, *q* out *n*, *q* over *n*, *q* since *n*, *q* unlike *n*	*q* behind *n*, *q* by *n*, *q* like *n*, *q* obj *n*, *n* sbj_intr *q*, *q* through *n*, *q* via *n*

Table 15
Average qualia extraction performance with automatically harvested links (compare to Table 11).

model	*precision*	*recall*	*F*	*F s.d.*
TypeDM	24.2	26.7	19.1	7.7
DepDM	18.4	27.0	15.1	4.9
LexDM	22.6	18.1	14.8	7.7

$L{\times}W_1W_2$ vectors might be useful for feature selection in $W_1W_2{\times}L$ or for tasks in which we are given a set of pairs, and we have to find links that can function as verbal labels for the relation between the word pairs (Turney 2006a).

6.5 Smoothing by Tensor Decomposition

Dimensionality reduction techniques such as the (truncated) SVD approximate a sparse co-occurrence matrix with a denser lower-rank matrix of the same size, and they have been shown to be effective in many semantic tasks, probably because they provide a beneficial form of smoothing of the dimensions. See Turney and Pantel (2010) for references and discussion. We can apply SVD (or similar methods) to any of the tensor-derived matrices we used for the tasks herein. An interesting alternative is to smooth the source tensor directly by a tensor decomposition technique. In this section, we present (very preliminary) evidence that tensor decomposition can improve performance, and it is at least as good in this respect as matrix-based SVD. This is the only experiment in which we operate on the tensor directly, rather than on the matrices derived from it, paving the way to a more active role for the underlying tensor in the DM approach to semantics.

The (truncated) Tucker decomposition of a tensor can be seen as a higher-order generalization of SVD. Given a tensor $\boldsymbol{\mathcal{X}}$ of dimensionality $I_1 \times I_2 \times I_3$, its n-rank R_n is the rank of the vector space spanned by its mode-n fibers (obviously, for each mode n of the tensor, $R_n \leq I_n$). Tucker decomposition approximates the tensor $\boldsymbol{\mathcal{X}}$ having n-ranks $R_1, \ldots, R_n$ with $\tilde{\boldsymbol{\mathcal{X}}}$, a tensor with n-ranks $Q_n \leq R_n$ for all modes n. Unlike the case of SVD, there is no analytical procedure to find the best lower-rank approximation to a tensor, and Tucker decomposition algorithms search for the reduced rank tensor with the best fit (as measured by least square error) iteratively. Specifically, we use the memory-efficient MET(1) algorithm of Kolda and Sun (2008) as implemented in the Matlab Tensor Toolbox.[10] Kolda and Bader (2009) provide details on Tucker decomposition, its general properties, as well as applications and alternatives.

SVD is believed to exploit patterns of higher order co-occurrence between the rows and columns of a matrix (Manning and Schütze 1999; Turney and Pantel 2010), making row elements that co-occur with two synonymic columns more similar than in the original space. Tucker decomposition applied to the mode-3 tuple tensor could capture patterns of higher order co-occurrence for each of the modes. For example, it might capture at the same time similarities between links such as *use* and *hold* and w_2 elements such as *gun* and *knife*. SVD applied after construction of the $W_1{\times}LW_2$ matrix, on the other hand, would miss the composite nature of columns such as $\langle$*use, gun*$\rangle$, $\langle$*use, knife*$\rangle$ and $\langle$*hold, gun*$\rangle$. Another attractive feature of Tucker decomposition is that it could be

10 `http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/`.

Table 16
Purity in Almuhareb–Poesio concept clustering with rank reduction of the APTypeDM tensor; 95% confidence intervals (CI) obtained by bootstrapping.

reduction	*rank*	*purity*	*95% CI*
Tucker	$250\times50\times500$	75	72–80
Tucker	$300\times50\times500$	75	71–79
Tucker	$300\times50\times450$	74	71–79
SVD	200	74	71–79
SVD	350	74	70–79
Tucker	$300\times40\times500$	74	70–78
Tucker	$300\times60\times500$	74	70–78
Tucker	$350\times50\times500$	73	69–77
Tucker	$300\times50\times550$	72	69–77
SVD	250	72	69–77
SVD	150	72	68–77
none	≤ 402	71	69–77
SVD	300	71	68–76
SVD	100	68	65–73
SVD	50	64	61–70

applied once to smooth the source tensor, whereas with SVD each matricization must be smoothed separately. However, Tucker decomposition and SVD are computationally intensive procedures, and, at least with our current computational resources, we are not able to decompose even the smallest DM tensor (similarly, we cannot apply SVD to a full matricization). Given the continuous growth in computational power and the fact that efficient tensor decomposition is a very active area of research (Turney 2007; Kolda and Sun 2008) full tensor decomposition is nevertheless a realistic near future task.

For the current pilot study, we replicated the AP concept clustering experiment described in Section 6.1.3. Because for efficiency reasons we must work with just a portion of the original tensor, we thought that the AP data set, consisting of a relatively large and balanced collection of nominal concepts, would offer a sensible starting point to extract the subset. Specifically, we extract from our best tensor TypeDM the values labeled by tuples $\langle w_{AP}, l, w_2\rangle$ where w_{AP} is in the AP set, l is one of the 100 most common links occurring in tuples with a w_{AP}, and w_2 is one of the 1,000 most common words occurring in tuples with a w_{AP} and a l. The resulting (sub-)tensor, APTypeDM, has dimensionality $402 \times 100 \times 1,000$ with 1,318,214 non-zero entries (density: 3%). The $W_1{\times}LW_2$ matricization of APTypeDM results in a $402 \times 1,000,000$ matrix with 66,026 non-zero columns and the same number of non-zero entries and density as the tensor.

The possible combinations of target lower n-ranks constitute a large tridimensional parameter space, and we leave its systematic exploration to further work. Instead, we pick 300, 50, and 500 as (intuitively reasonable) initial target n-ranks for the three modes, and we explore their neighborhood in parameter space by changing one target n-rank at a time, by a relatively small value (300 ± 50, 50 ± 10, and 500 ± 50, respectively). For the parameters concerning the reduced tensor fitting process, we accept the default values of the Tensor Toolbox. For comparison purposes, we also apply SVD to the $W_1{\times}LW_2$ matrix derived from APTypeDM. We systematically explore the SVD target lower rank parameter from 50 to 350 in increments of 50 units.

The results are reported in Table 16. The rank column reports the n-ranks when reduction is performed on the tensor, and matrix ranks in the other cases. Bootstrapped confidence intervals are obtained as described in Section 6.1.3. In general, the results

confirm that smoothing by rank reduction is beneficial to semantic performance, although not spectacularly so, with an improvement of about 4% for the best reduced model with respect to the raw APTypeDM tensor (consider however also the relatively wide confidence intervals). As a general trend, tensor-based smoothing (Tucker) does better than matrix-based smoothing (SVD). As we said, for Tucker we only report results from a small region of the tridimensional parameter space, whereas the SVD rank parameter range is explored coarsely but exhaustively. Thus, although other parameter combinations might lead to dramatic changes in Tucker performance, the best SVD performance in the table is probably close to the SVD performance upper bound.

The present pilot study suggests an attitude of cautious optimism towards tensor decomposition as a smoothing technique. At least in the AP task, it helps as compared to no smoothing at all. The same conclusion is reached by Turney (2007), who uses essentially the same method (with some differences in implementation) to tackle the TOEFL task, and obtains more than 10% improvement in accuracy with respect to the corresponding raw tensor. At least as a trend, tensor decomposition appears to be better than matrix decomposition, but only marginally so (Turney does not perform this comparison). Still, even if the tensor- and matrix-based decompositions turned out to have comparable effects, tensor-based smoothing is more attractive in the DM framework because we could perform the decomposition once, and use the smoothed tensor as our stable underlying DM (modulo, of course, memory problems with computing such a large tensor decomposition).

Beyond smoothing, tensor decomposition might provide some novel avenues for distributional semantics, while keeping to the DM program of a single model for many tasks. Van de Cruys (2009) used tensor decomposition to find commonalities in latent dimensions across the fiber labels (in the DM formalism, this would amount to finding commonalities across w_1, l, and w_2 elements). Another possible use for smoothing would be to propagate "link mass" across parts of speech. Our tensors, being based on POS tagging and dependency parsing, have 0 values for noun-link-noun tuples such as $\langle$*city, obj, destruction*$\rangle$ and $\langle$*city, subj_tr, destruction*$\rangle$. In a smoothed tensor, by the influence of tuples such as $\langle$*city, obj, destroy*$\rangle$ and $\langle$*city, sbj_tr, destroy*$\rangle$, these tuples will get some non-0 weight that, hopefully, will make the object relation between *city* and *destruction* emerge. This is at the moment just a conjecture, but it constitutes an exciting direction for further work focusing on tensor decomposition within the DM framework.

7. Conclusion

A general framework for distributional semantics should satisfy the following two requirements: (1) representing corpus-derived data in such a way as to capture aspects of meaning that have so far been modeled with different, prima facie incompatible data structures; (2) using this common representation to address a large battery of semantic experiments, achieving a performance at least comparable to that of state-of-art, task-specific DSMs. We can now safely claim that DM satisfies both these desiderata, and thereby represents a genuine step forward in the quest for a general purpose approach to distributional semantics.

DM addresses point (1) by modeling distributional data as a structure of weighted tuples that is formalized as a labeled third-order tensor. This is a generalization with respect to the common approach of many corpus-based semantic models (the structured DSMs) that rely on distributional information encoded into word–link–word tuples, associated with weights that are functions of their frequency of co-occurrence in the corpus. Existing structured DSMs still couch this information directly in binary structures,

namely, co-occurrence matrices, thereby giving rise to different semantic spaces and losing sight of the fact that such spaces share the same kind of distributional information. The third-order tensor formalization of distributional data allows DM to fully exploit the potential of corpus-derived tuples. The four semantic spaces we analyzed and tested in Section 6 are generated from the same underlying third-order tensor, by the standard operation of tensor matricization. This way, we derive a set of semantic spaces that can be used for measuring attributional similarity (finding synonyms, categorizing concepts into superordinates, etc.) and relational similarity (finding analogies, grouping concept pairs into relation classes, etc.). Moreover, the distributional information encoded in the tensor and unfolded via matricization leads to further arrangements of the data useful in addressing semantic problems that do not fall straightforwardly into the attributional or the relational paradigm (grouping verbs by alternations, harvesting patterns that represent a relation). In some cases, it is obvious how to reformulate a semantic problem in the new framework. Other tasks can be reframed in terms of our four semantic spaces using geometric operations such as centroid computations and projection onto a subspace. This was the case for selectional preferences, pattern- and example-based relation extraction (illustrated by qualia harvesting), and the task of generating typical properties of concepts. We consider a further strength of the DM approach that it naturally encourages us to think, as we did in these cases, of ways to tackle apparently unrelated tasks with the existing resources, rather than devising unrelated approaches to deal with them.

Regarding point (2), that is, addressing a large battery of semantic experiments with good performance, in nearly all test sets our best implementation of DM (TypeDM) is at least as good as other algorithms reported in recently published papers (typically developed or tuned for the task at hand), often towards (or at) the top of the state-of-the-art ranking. Where other models outperform TypeDM by a large margin, there are typically obvious reasons for this: The rivals have been trained on much larger corpora, or they rely on special knowledge resources, or on sophisticated machine learning algorithms. Importantly, TypeDM is consistently at least as good (or better than) those models we reimplemented to be fully comparable to our DMs (i.e., Win, DV, LRA).

Moreover, the best DM implementation does not depend on the semantic space: TypeDM outperforms (at least in terms of average performance across tasks) the other two models in all four spaces. This is not surprising (better distributional tuples should still be better when seen from different views), but it is good to have an empirical confirmation of the a priori intuition. The current results suggest that one could, for example, compare alternative DMs on a few attributional tasks, and expect the best DM in these tasks to also be the best in relational tasks and other semantic challenges.

The final experiment of Section 6 briefly explored an interesting aspect of the tensor-based formalism, namely, the possibility of improving performance on some tasks by working directly on the tensor (in this case, applying tensor rank reduction for smoothing purposes) rather than on the matrices derived from it. Besides this pilot study, we did not carry out any task-specific optimization of TypeDM, which achieves its very good performance using exactly the same underlying parameter configuration (e.g., dependency paths, weighting function) across the different spaces and tasks. Parameter tuning is an important aspect in DSM development, with an often dramatic impact of parameter variation (Bullinaria and Levy 2007; Erk and Padó 2009). We leave the exploration of parameter space in DM for future research. Its importance notwithstanding, however, we regard this as a rather secondary aspect, if compared with the good performance of a DM model (even in its current implementation) in the large and multifarious set of tasks we presented.

Of course, many issues are still open. It is one thing to claim that the models that outperform TypeDM do so because they rely on larger corpora; it is another to show that TypeDM trained on more data *does* reach the top of the current heap. The differences between TypeDM and the other, generally worse-performing DM models remind us that the idea of a shared distributional memory per se is not enough to obtain good results, and the extraction of an ideal DM from the corpus certainly demands further attention. We need to reach a better understanding of which pieces of distributional information to extract, and whether different semantic tasks require focusing on specific subsets of distributional data. Another issue we completely ignored but which will be of fundamental importance in applications is how a DM-based system can deal with out-of-vocabulary items. Ideally, we would like a seamless way to integrate new terms in the model incrementally, based on just a few extra data points, but we leave it to further research to study how this could be accomplished, together with the undoubtedly many further practical and theoretical problems that will emerge. We will conclude, instead, by discussing some general advantages that follow from the DM approach of separating corpus-based model building, the multi-purpose long term distributional memory, and different views of the memory data to accomplish different semantic tasks, without resorting to the source corpus again.

First of all, we would like to make a more general point regarding parameter tuning and task-specific optimization, by going back to the analogy with WordNet as a semantic multi-purpose resource. If you want to improve performance of a WordNet-based system, you will probably not wait for its next release, but rather improve the algorithms that work on the existing WordNet graph. Similarly, in the DM approach we propose that corpus-based resources for distributional semantics should be relatively stable, multi-purpose, large-scale databases (in the form of weighted tuple structures), only occasionally updated (because a better or larger corpus becomes available, a better parser, etc.). Still, given the same underlying DM and a certain task, much work can be done to exploit the DM optimally in the task, with no need to go back to corpus-based resource construction. For example, performance on attributional tasks could be raised by dimension reweighting techniques such as recently proposed by Zhitomirsky-Geffet and Dagan (2009). For the problem of data sparseness in the $W_1W_2{\times}L$ space, we could treat the tensor as a graph and explore random walks and other graphical approaches that have been shown to "scale down" gracefully to capture relations in sparser data sets (Minkov and Cohen 2007, 2008). As in our simple example of smoothing relational pairs with attributional neighbors, more complex tasks may be tackled by combining different views of DM, and/or resorting to different (sub)spaces within the same view, as in our approach to selectional preferences. One might even foresee an algorithmic way to mix and match the spaces as most appropriate to a certain task. We propose a similar split for the role of supervision in DSMs. Construction of the DM tensor from the corpus is most naturally framed as an unsupervised task, because the model will serve many different purposes. On the other hand, supervision can be of great help in tuning the DM data to specific tasks (as we did, in a rather naive way, with the nearest centroid approach to most non-attributional tasks). A crucial challenge for DSMs is whether and how corpus-derived vectors can also be used in the construction of meaning for constituents larger than words. These are the traditional domains of formal semantics, which is most interested in how the logical representation of a sentence or a discourse is built compositionally by combining the meanings of its constituents. DSMs have so far focused on representing lexical meaning, and compositional and logical issues have either remained out of the picture, or have received still unsatisfactory accounts. A general consensus exists on the need to overcome this limitation, and to build new bridges

between corpus-based semantics and symbolic models of meanings (Clark and Pulman 2007; Widdows 2008). Most problems encountered by DSMs in tackling this challenge are specific instances of more general issues concerning the possibility of representing symbolic operations with distributed, vector-based data structures (Markman 1999). Many avenues are currently being explored in corpus-based semantics, and interesting synergies are emerging with research areas such as neural systems (Smolensky 1990; Smolensky and Legendre 2006), quantum information (Widdows and Peters 2003; Aerts and Czachor 2004; Widdows 2004; Van Rijsbergen 2004; Bruza and Cole 2005; Hou and Song 2009), holographic models of memory (Jones and Mewhort 2007), and so on. A core problem in dealing with compositionality with DSMs is to account for the role of syntactic information in determining the way semantic representations are built from lexical items. For instance, the semantic representation assigned to *The dog bites the man* must be different from the one assigned to *The man bites the dog*, even if they contain exactly the same lexical items. Although it is still unclear which is the best way to compose the representation of content words in vector spaces, it is nowadays widely assumed that structured representations like those adopted by DM are in the right direction towards a solution to this issue, exactly because they allow distributional representations to become sensitive to syntactic structures (Erk and Padó 2008). Compositionality and similar issues in DSMs lie beyond the scope of this paper. However, there is nothing in DM that prevents it from interacting with any of the research directions we have mentioned here. Indeed, we believe that the generalized nature of DM represents a precondition for distributional semantics to be able to satisfactorily address these more advanced challenges. A multi-purpose, distributional semantic resource like DM can allow researchers to focus on the next steps of semantic modeling. These include compositionality, but also modulating word meaning in context (Erk and Padó 2008; Mitchell and Lapata 2008) and finding ways to embed the distributional memory in complex NLP systems (e.g., for question answering or textual entailment) or even embodied agents and robots.

DM-style triples predicating a relation between two entities are common currency in many semantic representation models (e.g., semantic networks) and knowledge-exchange formalisms such as RDF. This might also pave the way to the integration of corpus-based information with other knowledge sources. It is hard to see how such integration could be pursued within generalized systems, such as PairClass (Turney 2008), that require keeping a full corpus around and corpus-processing know-how on behalf of interested researchers from outside the NLP community (see discussion in Section 4 above). Similarly, the DM triples might help in fostering the dialogue between computational linguists and the computational neuro-cognitive community, where it is common to adopt triple-based representations of knowledge, and to use the same set of tuples to simulate various aspects of cognition. For a recent extended example of this approach, see Rogers and McClelland (2004). It would be relatively easy to use a DM model in lieu of their neural network, and use it to simulate the conceptual processes they reproduce.

DM, unlike classic DSM models that go directly from the corpus data to solving specific semantic tasks, introduces a clear distinction between an acquisition phase (corpus-based tuple extraction and weighting), the declarative structure at the core of semantic modeling (the distributional memory), and the procedural problem-solving components (possibly supervised procedures to perform different semantic tasks). This separation is in line with what is commonly assumed in cognitive science and formal linguistics, and we hope it will contribute to make corpus-based modeling a core part of the ongoing study of semantic knowledge in humans and machines.

Acknowledgments

We thank Abdulrahman Almuhareb, Philipp Cimiano, George Karypis, Tamara Kolda, Thomas Landauer, Mirella Lapata, Ken McRae, Brian Murphy, Vivi Nastase, Diarmuid Ó Séaghdha, Sebastian and Ulrike Padó, Suzanne Stevenson, Peter Turney, their colleagues, and the SEMEVAL Task 4 organizers for data and tools. We thank Gemma Boleda, Phillipp Cimiano, Katrin Erk, Stefan Evert, Brian Murphy, Massimo Poesio, Magnus Sahlgren, Tim Van de Cruys, Peter Turney, and three anonymous reviewers for a mixture of advice, clarification, and ideas.

References

Aerts, Diederik and Marek Czachor. 2004. Quantum aspects of semantic analysis and symbolic artificial intelligence. *Journal of Physics A: Mathematical and General*, 37:123–132.

Alishahi, Afra and Suzanne Stevenson. 2008. A distributional account of the semantics of multiword expressions. *Italian Journal of Linguistics*, 20(1):157–179.

Almuhareb, Abdulrahman. 2006. *Attributes in Lexical Acquisition*. Ph.D. thesis, University of Essex.

Almuhareb, Abdulrahman and Massimo Poesio. 2004. Attribute-based and value-based clustering: An evaluation. In *Proceedings of EMNLP*, pages 158–165, Barcelona.

Almuhareb, Abdulrahman and Massimo Poesio. 2005. Concept learning and categorization from the web. In *Proceedings of CogSci*, pages 103–108, Stresa.

Baldwin, Timothy, Valia Kordoni, and Aline Villavicencio. 2009. Prepositions in applications: A survey and introduction to the special issue. *Computational Linguistics*, 35(2):119–149.

Baroni, Marco, Eduard Barbu, Brian Murphy, and Massimo Poesio. 2010. Strudel: A distributional semantic model based on properties and types. *Cognitive Science*, 34(2):222–254.

Baroni, Marco, Stefan Evert, and Alessandro Lenci, editors. 2008. *Bridging the Gap between Semantic Theory and Computational Simulations: Proceedings of the ESSLLI Workshop on Distributional Lexical Semantic*. FOLLI, Hamburg.

Baroni, Marco and Alessandro Lenci. 2008. Concepts and properties in word spaces. *Italian Journal of Linguistics*, 20(1):55–88.

Baroni, Marco and Alessandro Lenci. 2009. One distributional memory, many semantic tasks. In *Proceedings of the EACL GEMS Workshop*, pages 1–8, Athens.

Biciçi, Ergun and Deniz Yuret. 2006. Clustering word pairs to answer analogy questions. In *Proceedings of the Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 277–284, Muğla.

Bruza, Peter and Richard Cole. 2005. Quantum logic of semantic space: An exploratory investigation of context effects in practical reasoning. In Sergei Artemov, Howard Barringer, Arthur d'Avila Garcez, Luis C. Lamb, and John Woods, editors, *We Will Show Them: Essays in Honour of Dov Gabbay, volume one*. College Publications, London, pages 339–361.

Buitelaar, Paul, Philipp Cimiano, and Bernardo Magnini. 2005. *Ontology Learning from Text*. IOS Press, Amsterdam.

Bullinaria, John and Joseph Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

Chen, Hsin-Hsi, Ming-Shun Lin, and Yu-Chuan Wei. 2006. Novel association measures using Web search with double checking. In *Proceedings of COLING-ACL*, pages 1009–1016, Sydney.

Church, Kenneth and Peter Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Cimiano, Philipp and Johanna Wenderoth. 2007. Automatic acquisition of ranked qualia structures from the Web. In *Proceedings of ACL*, pages 888–895, Prague.

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

Curran, James and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the ACL Workshop on Unsupervised Lexical Acquisition*, pages 59–66, Philadelphia, PA.

Davidov, Dmitry and Ari Rappoport. 2008a. Classification of semantic relationships between nominals using pattern clusters. In *Proceedings of ACL*, pages 227–235, Columbus, OH.

Davidov, Dmitry and Ari Rappoport. 2008b. Unsupervised discovery of

generic relationships using pattern clusters and its evaluation by automatically generated SAT analogy questions. In *Proceedings of ACL*, pages 692–700, Columbus, OH.
Dietterich, Thomas. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.
Dowty, David. 1977. *Word Meaning and Montague Grammar*. Kluwer, Dordrecht.
Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.
Efron, Bradley and Robert Tibshirani. 1994. *An Introduction to the Bootstrap*. Chapman and Hall, Boca Raton, FL.
Erk, Katrin. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of ACL*, pages 216–223, Prague.
Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, HI.
Erk, Katrin and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the EACL GEMS Workshop*, pages 57–65, Athens.
Evert, Stefan. 2005. *The Statistics of Word Cooccurrences*. Ph.D. dissertation, Stuttgart University.
Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
Garrard, Peter, Matthew Lambon Ralph, John Hodges, and Karalyn Patterson. 2001. Prototypicality, distinctiveness, and intercorrelation: Analyses of the semantic attributes of living and nonliving concepts. *Cognitive Neuropsychology*, 18(2):25–174.
Geeraerts, Dirk. 2010. *Theories of Lexical Semantics*. Oxford University Press, Oxford.
Girju, Roxana, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1):83–135.
Girju, Roxana, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of SemEval 2007*, pages 13–18, Prague.
Grefenstette, Gregory. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer, Boston, MA.
Griffiths, Tom, Mark Steyvers, and Josh Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114:211–244.
Harris, Zellig. 1954. Distributional structure. *Word*, 10(2-3):1456–1162.
Hearst, Marti. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, pages 539–545, Nantes.
Hearst, Marti. 1998. Automated discovery of WordNet relations. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, pages 131–151.
Herdağdelen, Amaç and Marco Baroni. 2009. BagPack: A general framework to represent semantic relations. In *Proceedings of the EACL GEMS Workshop*, pages 33–40, Athens.
Herdağdelen, Amaç, Katrin Erk, and Marco Baroni. 2009. Measuring semantic relatedness with vector space models and random walks. In *Proceedings of TextGraphs-4*, pages 50–53, Singapore.
Heylen, Kris, Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2008. Modelling word similarity: An evaluation of automatic synonymy extraction algorithms. In *Proceedings of LREC*, pages 3243–3249, Marrakech.
Hou, Yuexian and Dawei Song. 2009. Characterizing pure high-order entanglements in lexical semantic spaces via information geometry. In Peter Bruza, Donald Sofge, William Lawless, and C. J. van Rijsbergen, editors, *Quantum Interaction: Third International Symposium, QI 2009*. Springer, Berlin, pages 237–250.
Jackendoff, Ray. 1990. *Semantic Structures*. MIT Press, Cambridge, MA.
Joanis, Eric, Suzanne Stevenson, and David James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.
Jones, Michael and Douglas Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.
Karypis, George. 2003. CLUTO: A clustering toolkit. Technical Report 02-017, University of Minnesota Department of Computer Science, Minneapolis.
Kilgarriff, Adam, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The Sketch Engine. In *Proceedings of Euralex*, pages 105–116, Lorient.

Kolda, Tamara. 2006. Multilinear operators for higher-order decompositions. Technical Report 2081, SANDIA, Albuquerque, NM.
Kolda, Tamara and Brett Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.
Kolda, Tamara and Jimeng Sun. 2008. Scalable tensor decompositions for multi-aspect data mining. In *Proceedings of ICDM*, pages 94–101, Pisa.
Landauer, Thomas and Susan Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.
Lenci, Alessandro. 2008. Distributional approaches in linguistic and cognitive research. *Italian Journal of Linguistics*, 20(1):1–31.
Lenci, Alessandro. 2010. The life cycle of knowledge. In Chu-Ren Huang, Nicoletta Calzolari, Aldo Gangemi, Alessandro Lenci, Alessandro Oltramari, and Laurent Prévot, editors, *Ontology and the Lexicon. A Natural Language Processing Perspective*. Cambridge University Press, Cambridge, UK, pages 241–257.
Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.
Levin, Beth and Malka Rappaport-Hovav. 2005. *Argument Realization*. Cambridge University Press, Cambridge, UK.
Lin, Dekang. 1998a. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*, pages 768–774, Montreal.
Lin, Dekang. 1998b. An information-theoretic definition of similarity. In *Proceedings of ICML*, pages 296–304, Madison, WI.
Liu, Hugo and Push Singh. 2004. ConceptNet: A practical commonsense reasoning toolkit. *BT Technology Journal*, pages 211–226.
Lowe, Will. 2001. Towards a theory of semantic space. In *Proceedings of CogSci*, pages 576–581, Edinburgh, UK.
Lund, Kevin and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28:203–208.
Manning, Chris and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.
Markman, Arthur B. 1999. *Knowledge Representation*. Psychology Press, New York, NY.
Matveeva, Irina, Gina-Anne Levow, Ayman Farahat, and Christian Royer. 2005. Generalized latent semantic analysis for term representation. In *Proceedings of RANLP*, pages 60–68, Borovets.
McRae, Ken, George Cree, Mark Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559.
McRae, Ken, Michael Spivey-Knowlton, and Michael Tanenhaus. 1998. Modeling the influence of thematic fit (and other constraints) in on-line sentence comprehension. *Journal of Memory and Language*, 38:283–312.
Merlo, Paola and Suzanne Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.
Meyer, Carl. 2000. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA.
Miller, George and Walter Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6:1–28.
Minkov, Einat and William Cohen. 2007. Learning to rank typed graph walks: Local and global approaches. In *Proceedings of WebKDD/SNA-KDD*, pages 1–8, San José, CA.
Minkov, Einat and William Cohen. 2008. Learning graph walk based similarity measures for parsed text. In *Proceedings of EMNLP*, pages 907–916, Honolulu, HI.
Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.
Murphy, Gregory. 2002. *The Big Book of Concepts*. MIT Press, Cambridge, MA.
Nastase, Vivi and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proceedings of the Fifth International Workshop on Computational Semantics*, pages 285–301, Tilburg, The Netherlands.
Ó Séaghdha, Diarmuid and Ann Copestake. 2009. Using lexical and relational similarity to classify semantic relations. In *Proceedings of EACL*, pages 621–629, Athens.
Padó, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.
Padó, Ulrike. 2007. *The Integration of Syntax and Semantic Plausibility in a*

Wide-Coverage Model of Sentence Processing. Ph.D. dissertation, Saarland University, Saarbrücken.
Padó, Ulrike, Sebastian Padó, and Katrin Erk. 2007. Flexible, corpus-based modelling of human plausibility judgements. In *Proceedings of EMNLP*, pages 400–409, Prague.
Pantel, Patrick and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of COLING-ACL*, pages 113–120, Sydney.
Peirsman, Yves and Dirk Speelman. 2009. Word space models of lexical variation. In *Proceedings of the EACL GEMS Workshop*, pages 9–16, Athens.
Pustejovsky, James. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.
Quesada, Jose, Praful Mangalath, and Walter Kintsch. 2004. Analogy-making as predication using relational information and LSA vectors. In *Proceedings of CogSci*, page 1623, Chicago, IL.
Raghunathan, Trivellore. 2003. An approximate test for homogeneity of correlated correlation coefficients. *Quality & Quantity*, 37:99–110.
Rapp, Reinhard. 2003. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the 9th MT Summit*, pages 315–322, New Orleans, LA.
Rapp, Reinhard. 2004. A freely available automatically generated thesaurus of related words. In *Proceedings of LREC*, pages 395–398, Lisbon.
Rogers, Timothy and James McClelland. 2004. *Semantic Cognition: A Parallel Distributed Processing Approach*. MIT Press, Cambridge, MA.
Rothenhäusler, Klaus and Hinrich Schütze. 2009. Unsupervised classification with dependency based word spaces. In *Proceedings of the EACL GEMS Workshop*, pages 17–24, Athens, Greece.
Rubenstein, Herbert and John Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.
Ruiz-Casado, Maria, Enrique Alfonseca, and Pablo Castells. 2005. Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of RANLP*, pages 1–7, Borovets.
Sagi, Eyal, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the EACL GEMS Workshop*, pages 104–111, Athens.
Sahlgren, Magnus. 2005. An introduction to random indexing. `http://www.sics.se/~mange/papers/RI_intro.pdf`.
Sahlgren, Magnus. 2006. *The Word-Space Model*. Ph.D. dissertation, Stockholm University.
Schulte im Walde, Sabine. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32:159–194.
Schütze, Hinrich. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI Publications, Stanford, CA.
Smolensky, Paul. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.
Smolensky, Paul and Geraldine Legendre. 2006. *The Harmonic Mind. From Neural Computation to Optimality-theoretic Grammar*. MIT Press, Cambridge, MA.
Terra, Egidio and Charles Clarke. 2003. Frequency estimates for statistical word similarity measures. In *Proceedings of HLT-NAACL*, pages 244–251, Edmonton.
Turney, Peter. 2001. Mining the Web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML*, pages 491–502, Freiburg.
Turney, Peter. 2006a. Expressing implicit semantic relations without supervision. In *Proceedings of COLING-ACL*, pages 313–320, Sydney.
Turney, Peter. 2006b. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.
Turney, Peter. 2007. Empirical evaluation of four tensor decomposition algorithms. Technical Report ERB-1152, NRC, Ottawa.
Turney, Peter. 2008. A uniform approach to analogies, synonyms, antonyms and associations. In *Proceedings of COLING*, pages 905–912, Manchester.
Turney, Peter and Michael Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278.
Turney, Peter and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.
Van de Cruys, Tim. 2009. A non-negative tensor factorization model for selectional preference induction. In *Proceedings of the EACL GEMS Workshop*, pages 83–90, Athens.
Van Overschelde, James, Katherine Rawson, and John Dunlosky. 2004. Category

norms: An updated and expanded version of the Battig and Montague (1969) norms. *Journal of Memory and Language*, 50:289–335.
Van Rijsbergen, C. J. 2004. *The Geometry of Information Retrieval*. Cambridge University Press, Cambridge, UK.
Veale, Tony and Yanfen Hao. 2008. Acquiring naturalistic concept descriptions from the Web. In *Proceedings of LREC*, pages 1121–1124, Marrakech.
Vinson, David and Gabriella Vigliocco. 2008. Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, 40(1):183–190.
Widdows, Dominic. 2004. *Geometry and Meaning*. CSLI Publications, Stanford, CA.
Widdows, Dominic. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*, pages 1–8, Oxford.
Widdows, Dominic and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *Proceedings of ICCL*, pages 1–7, Taipei.
Widdows, Dominic and Stanley Peters. 2003. Word vectors and quantum logic. In *Proceedings of the Eighth Mathematics of Language Conference*, pages 1–14, Bloomington, IN.
Zarcone, Alessandra and Alessandro Lenci. 2008. Computational models of event type classification in context. In *Proceedings of LREC*, pages 1232–1238, Marrakech.
Zhao, Ying and George Karypis. 2003. Criterion functions for document clustering: Experiments and analysis. Technical Report 01-40, University of Minnesota Department of Computer Science, Minneapolis.
Zhitomirsky-Geffet, Maayan and Ido Dagan. 2009. Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35(3):435–461.