Deriving Boolean structures from distributional vectors

German Kruszewski **Denis Paperno** **Marco Baroni**

Center for Mind/Brain Sciences
University of Trento
firstname.lastname@unitn.it

Abstract

Corpus-based distributional semantic models capture degrees of semantic relatedness among the words of very large vocabularies, but have problems with logical phenomena such as entailment, that are instead elegantly handled by model-theoretic approaches, which, in turn, do not scale up. We combine the advantages of the two views by inducing a mapping from distributional vectors of words (or sentences) into a Boolean structure of the kind in which natural language terms are assumed to denote. We evaluate this Boolean Distributional Semantic Model (BDSM) on recognizing entailment between words and sentences. The method achieves results comparable to a state-of-the-art SVM, degrades more gracefully when less training data are available and displays interesting qualitative properties.

1 Introduction

Different aspects of natural language semantics have been studied from different perspectives. Distributional semantic models (Turney and Pantel, 2010) induce large-scale vector-based lexical semantic representations from statistical patterns of word usage. These models have proven successful in tasks relying on meaning relatedness, such as synonymy detection (Landauer and Dumais, 1997), word sense discrimination (Schütze, 1997), or even measuring phrase plausibility (Vecchi et al., 2011). On the other hand, logical relations and operations, such as entailment, contradiction, conjunction and negation, receive an elegant treatment in formal semantic models. The latter lack, however, general procedures to learn from data, and consequently have problems scaling up to real-life problems.

Formal semantics captures fundamental aspects of meaning in set-theoretic terms: Entailment, for example, is captured as the *inclusion* relation between the sets (of the relevant type) denoted by words or other linguistic expressions, e.g., sets of possible worlds that two propositions hold of (Chierchia and McConnell-Ginet, 2000, 299). In finite models, a mathematically convenient way to represent these denotations is to encode them in *Boolean* vectors, i.e., vectors of 0s and 1s (Sasao, 1999, 21). Given all elements e_i in the domain in which linguistic expressions of a certain type denote, the Boolean vector associated to a linguistic expression of that type has 1 in position i if $e_i \in S$ for S the set denoted by the expression, 0 otherwise. An expression a entailing b will have a Boolean vector including the one of b, in the sense that all positions occupied by 1s in the b vector are also set to 1 in the a vector. Very general expressions (entailing nearly everything else) will have very dense vectors, whereas very specific expressions will have very sparse vectors. The negation of an expression a will denote a "flipped" version of the a Boolean vector. *Vice versa*, two expressions with at least partially compatible meanings will have some overlap of the 1s in their vectors; conjunction and disjunction are carried through with the obvious bit-wise operations, etc.

To narrow the gap between the large-scale inductive properties of distributional semantic models and the logical power of Boolean semantics, we create Boolean meaning representations that build on the wealth of information inherent in distributional vectors of words (and sentences). More precisely, we use word (or sentence) pairs labeled as entailing

Transactions of the Association for Computational Linguistics, vol. 3, pp. 375–388, 2015. Action Editor: Alexander Koller.
Submission batch: 4/2015; Published 6/2015.

or not entailing to train a mapping from their distributional representations to Boolean vectors, enforcing feature inclusion in Boolean space for the entailing pairs. By focusing on inducing Boolean representations that respect the inclusion relation, our method is radically different from recent supervised approaches that learn an entailment classifier directly on distributional vectors, without enforcing inclusion or other representational constraints. We show, experimentally, that the method is competitive against state-of-the-art techniques in lexical entailment, improving on them in sentential entailment, while learning more effectively from less training data. This is crucial for practical applications that involve bigger and more diverse data than the focused test sets we used for testing. Moreover, extensive qualitative analysis reveals several interesting properties of the Boolean vectors we induce, suggesting that they are representations of greater generality beyond entailment, that might be exploited in further work for other logic-related semantic tasks.

2 Related work

Entailment in distributional semantics Due to the lack of methods to induce the relevant representations on the large scale needed for practical tasks, the Boolean structure defined by the entailment relation is typically not considered in efforts to automatically recognize entailment between words or sentences (Dagan et al., 2009). On the other hand, some researchers relying on distributional representations of meaning have attempted to apply various versions of the notion of feature inclusion to entailment detection. This is based on the intuitive idea – the so-called *distributional inclusion hypothesis* – that the features (vector dimensions) of a hypernym and a hyponym should be in a superset-subset relation, analogously to what we are trying to achieve in the Boolean space we induce, but directly applied to distributional vectors (Geffet and Dagan, 2005; Kotlerman et al., 2010; Lenci and Benotto, 2012; Weeds et al., 2004). It has been noticed that distributional context inclusion defines a Boolean structure on vectors just as entailment defines a Boolean structure on formal semantic representations (Clarke, 2012). However, the match between context inclusion and entailment is far from perfect.

First, distributional vectors are real-valued and contain way more nuanced information than simply inclusion or exclusion of certain features. Second, and more fundamentally, the information encoded in distributional vectors is simply not of the right kind since "feature inclusion" for distributional vectors boils down to contextual inclusion, and there is no reason to think that a hypernym should occur in all the contexts in which its hyponyms appear. For example, *bark* can be a typical context for *dog*, but we don't expect to find it a significant number of times with *mammal* even in a very large corpus. In practice distributional inclusion turns out to be a weak tool for recognizing the entailment relation (Erk, 2009; Santus et al., 2014) because denotational and distributional inclusion are independent properties.

More recently, several authors have explored supervised methods. In particular, Baroni et al. (2012), Roller et al. (2014) and Weeds et al. (2014) show that a Support Vector Machine trained on the distributional vectors of entailing or non-entailing pairs outperform the distributional inclusion measures. In our experiments, we will use this method as the main comparison point. The similarly supervised approach of Turney and Mohammad (2014) assumes the representational framework of Turney (2012), and we do not attempt to re-implement it here.

Very recently, other properties of distributional vectors, such as entropy (Santus et al., 2014) and topical coherence (Rimell, 2014), have been proposed as entailment cues. Since they are not based on feature inclusion, we see them as complementary, rather than alternative to our proposal.

Formal and distributional semantic models We try to derive a structured representation inspired by formal semantic theories from data-driven distributional semantic models. Combining the two approaches has proven a hard task. Some systems adopt logic-based representations but use distributional evidence for predicate disambiguation (Lewis and Steedman, 2013) or to weight probabilistic inference rules (Garrette et al., 2013; Beltagy et al., 2013). Other authors propose ways to encode aspects of logic-based representations such as logical connectives and truth values (Grefenstette, 2013) or predicate-argument structure (Clark and Pulman, 2007) in a vector-based framework. These studies

are, however, entirely theoretical. Rocktäschel et al. (2015) expand on the first, allowing for some generalization to unseen knowledge, by introducing some degree of fuzziness into the representations of predicates and terms. Still, this work does not attempt to map concepts to a logic-based representation nor tries to exploit the wealth of information contained in distributional vectors.

Socher et al. (2013), Bordes et al. (2012) and Jenatton et al. (2012) try to discover unseen facts from a knowledge base, which can be seen as a form of inference based on a restricted predicate logic. To do so, they build vector representations for entities, while relations are represented through classifiers. Only Socher et al. (2013) harness distributional vectors, and just as initialization values. The others, unlike us, do not build on independently-motivated word representations. Moreover, since the representations are learned from entities present in their knowledge base, one cannot infer the properties of unseen concepts.

In the spirit of inducing a variety of logical relations and operators (including entailment), Bowman (2013) applies a softmax classifier to the combined distributional representation of two given statements, which are in turn learned compositionally in a supervised fashion in order to guess the relation between them. The paper, however, only evaluates the model on a small restricted dataset, and it is unclear whether the method would scale to real-world challenges.

None of the papers with concrete implementations reviewed above tries, like us, to learn a Boolean structure where entailment corresponds to inclusion. A paper that does attempt to exploit a similar idea is Young et al. (2014), which also uses the notion of *model* from Formal Semantics to recognize entailment based on denotations of words and phrases. However, since the denotations in their approach are ultimately derived from human-generated captions of images, the method does not generalize to concepts that are not exemplified in the training database.

Finally, a number of studies, both theoretical (Baroni et al., 2014a; Coecke et al., 2010) and empirical (Paperno et al., 2014; Polajnar et al., 2014), adapt compositional methods from formal semantics to distributional vectors, in order to derive representations of phrases and sentences. This line of research applies formal operations to distributional representations, whereas we derive formal-semantics-like *representations* from distributional ones. Below, we apply our method to input sentence vectors constructed with the composition algorithm of Paperno et al. (2014).

3 The Boolean Distributional Semantic Model

We build the Boolean Distributional Semantic Model (BDSM) by mapping real-valued vectors from a distributional semantic model into Boolean-valued vectors, so that feature inclusion in Boolean space corresponds to entailment between words (or sentences). That is, we optimize the mapping function so that, if two words (or sentences) entail each other, then the more specific one will get a Boolean vector included in the Boolean vector of the more general one. The is illustrated in Figure 1.

Our model differs crucially from a neural network with a softmax objective in imposing a strong bias on the hypothesis space that it explores. In contrast to the latter, it only learns the weights corresponding to the mapping, while all other operations in the network (in particular, the inference step) are fixed in advance. The goal of such a bias is to improve learning efficiency and generalization using prior knowledge of the relation that the model must capture.

We will now discuss how the model is formalized in an incremental manner. The goal of the model is to find a function M_Θ (with parameters Θ) that maps the distributional representations into the Boolean vector space. To facilitate optimization, we relax the image of this mapping to be the full $[0, 1]$ interval, thus defining $M_\Theta : \mathbb{R}^N \mapsto [0, 1]^H$. This mapping has to respect the following condition as closely as possible: For two given words (or other linguistic expressions) p and q, and their distributional vectors v_p and v_q, all the active features (i.e., those having value close to 1) of $M_\Theta(v_p)$ must also be active in $M_\Theta(v_q)$ if and only if $p \Rightarrow q$.

To find such a mapping, we assume training data in the form of a sequence $[(p_k, q_k), y_k]_{k=1}^m$ containing both positive ($p_k \Rightarrow q_k$ and $y_k = 1$) and negative pairs ($p_k \not\Rightarrow q_k$ and $y_k = 0$). We learn the mapping by minimizing the difference between the model's

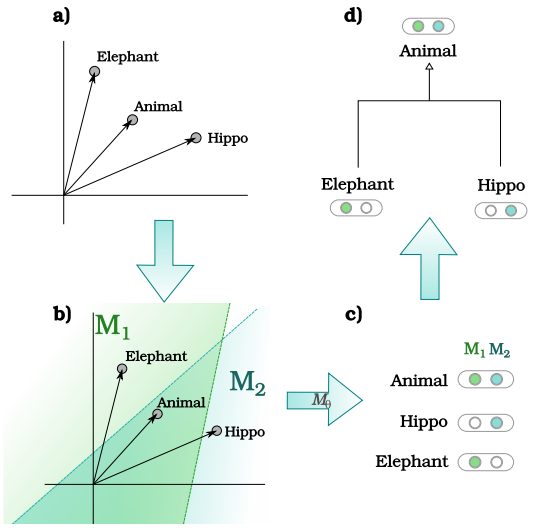

Figure 1: The BDSM architecture. a) Input distributional space b) Training of a Mapping M where each output dimension M_i can be seen as a (linear) cut in the original distributional space. c) Output representations after mapping. d) Fragment of the Boolean structure with example output representations.

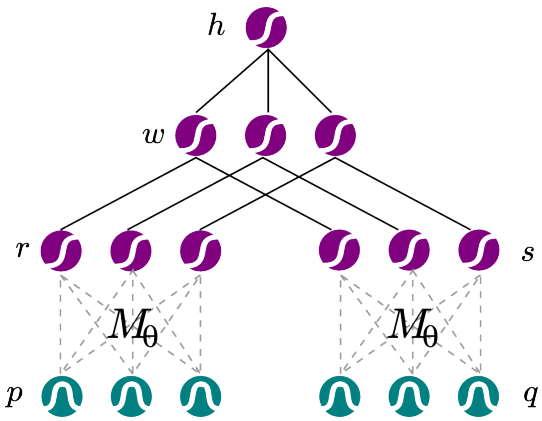

Figure 2: Schematic view of the entailment hypothesis function h_Θ. Solid links represent calculations that are fixed during learning, while dashed links represent the parameters Θ, which are being learned. The p and q input distributional vectors corresponding to each data point are fixed, r and s are their respective mapped Boolean representations. The w layer is a feature-inclusion detector and h is the final entailment judgment produced by the network.

entailment predictions (given by a function h_Θ) and the training targets, as measured by the MSE:

$$J(\Theta) = \frac{1}{2}\sum_{k=1}^{m}\left(h_\Theta(p_k, q_k) - y_k\right)^2 \qquad (1)$$

Here, we define M_Θ as a sigmoid function applied to a linear transformation: $M_\Theta(x) = g(Wx+b)$ and $g(x) = \frac{1}{1+e^{\frac{-x}{t}}}$, where t stands for an extra "temperature" parameter. We represent the $W \in \mathbb{R}^{H\times N}$, $b \in \mathbb{R}^H$ parameters succinctly by $\Theta = [W, b]$.

The calculation of $h_\Theta(p, q)$ involves a series of steps that can be construed as the architecture of a neural network, schematically represented in Figure 2. Recall that the output value of this function represents the model's prediction of the truth value for $p_k \Rightarrow q_k$. Here is an outline of how it is calculated. For each pair of words (or sentences) (p, q) in the training set, we map them onto their (soft) boolean correlates (r, s) by applying M_Θ to their corresponding distributional vectors. Next, we measure whether features that are active in r are also active in s (analogously to how Boolean implication works), obtaining a soft Boolean vector w. Finally, the output of h can be close to 1 only if all values in w are also close to 1. Thus, we compute the output value of h as the conjunction across all dimensions in w.

Concretely, $h_\Theta(p, q)$ is obtained as follows. The passage from the first to the second layer is computed as $r_\Theta = M_\Theta(v_p)$ and $s_\Theta = M_\Theta(v_q)$. Next, we compute whether the features that are active in r_Θ are also active in s_Θ. Given that we are working in the $[0, 1]$ range, we approximate this operation as $w_{\Theta i} = \max(1 - r_{\Theta i}, s_{\Theta i})$[1]. It is easy to see that if $r_{\Theta i} = 0$, then $w_{\Theta i} = 1$. Otherwise, $s_{\Theta i}$ must also be equal to 1 for $w_{\Theta i}$ to be 1. Finally, we compute $h_\Theta = \min_i w_{\Theta i}$. This is a way to compute the conjunction over the whole previous layer, thus checking whether all the features of r_Θ are included in those of s_Θ[2].

Finally, to allow for better generalization, the cost function is extended with two more components. The fist one is a L2 regularization term weighted by

[1] In practice, we use a differentiable approximation given by $\max(x, y) \approx \frac{\log(e^{Lx}+e^{Ly})}{L}$, where L is a sufficiently large number. We set $L = 100$, which yields results accurate enough for our purposes.

[2] Analogously, we use the differentiable approximation given by $min(w_\theta) = -\log(\frac{\sum_i e^{-Lw_{\theta i}}}{L})$

a parameter λ. The second one is a term that enforces sparsity of the resulting representations based on some desired level ρ.

3.1 Assessing entailment with BDSM

During training, positive pairs $p \Rightarrow q$ are required to satisfy full feature inclusion in their mapped representations (all the active features of $M_\Theta(v_p)$ must also be in $M_\Theta(v_q)$). At test time, we relax this condition to grant the model some flexibility. Concretely, entailment is quantified by the BI ("Boolean Inclusion") function, counting the proportion of features in the antecedent that are also present in the consequent after binarizing the outputs:

$$BI(u,v) = \frac{\sum_i \mathrm{rnd}(M_\Theta(u)_i)\,\mathrm{rnd}(M_\Theta(v)_i)}{\sum_i \mathrm{rnd}(M_\Theta(u)_i)}$$

where $\mathrm{rnd}(x) = \mathbb{1}[x > 0.5]$. The 0.5 threshold comes from construing each of the features in the output of M as probabilities. Of course, other formulas could be used to quantify entailment through BDSM, but we leave this to further research.

Since BI returns continuous values, we use development data to calculate a threshold e above which an entailment response is returned.

4 Evaluation setup

4.1 Distributional semantic spaces

Our approach is agnostic to the kind of distributional representation used, since it doesn't modify the input vectors, but builds on top of them. Still, it is interesting to test whether specific kinds of distributional vectors are better suited to act as input to BDSM. For our experiments, we use both the **count** and **predict** distributional semantic vectors of Baroni et al. (2014b).[3] These vectors were shown by their creators to reach the best average performance (among comparable alternatives) on a variety of semantic relatedness/similarity tasks, such as synonymy detection, concept categorization and analogy solving. If the same vectors turn out to also serve as good inputs for constructing Boolean representations, we are thus getting the best of both worlds: distributional vectors with proven high performance on relatedness/similarity tasks which can be mapped into a Boolean space to tackle logic-related tasks. We also experiment with the pre-trained vectors from **TypeDM** (Baroni and Lenci, 2010),[4] which are built by exploiting syntactic information, and should have different qualitative properties from the window-based approaches.

The count vectors of Baroni and colleagues are built from a 2-word-window co-occurrence matrix of 300k lower-cased words extracted from a 2.8 billion tokens corpus. The matrix is weighted using positive Pointwise Mutual Information (Church and Hanks, 1990). We use the full 300k×300k positive PMI matrix to compute the asymmetric similarity measures discussed in the next section, since the latter are designed for non-negative, sparse, full-rank representations. Due to efficiency constraints, for BDSM and SVM (also presented next), the matrix is reduced to 300 dimensions by Singular Value Decomposition (Schütze, 1997). The experiments of Baroni et al. (2014b) with these very same vectors suggest that SVD is *lowering* performance somewhat. So we are, if anything, giving an advantage to the simple asymmetric measures.

The predict vectors are built with the word2vec tool (Mikolov et al., 2013) on the same corpus and for the same vocabulary as the count vectors, using the CBOW method. They are constructed by associating 400-dimensional vectors to each word in the vocabulary and optimizing a single-layer neural network that, while traversing the training corpus, tries to predict the word in the center of a 5-word window from the vectors of those surrounding it. The word2vec subsampling parameter (that down-weights the impact of frequent words) is set to $1e^{-5}$.

Finally, TypeDM vectors were induced from the same corpus by taking into account the dependency links of a word with its sentential collocates. See Baroni and Lenci (2010) for details.

Composition methods For sentence entailment (Section 6), we need vectors for sentences, rather than words. We derive them from the count vectors compositionally in two different ways.[5] First, we use the additive model (**add**), under which we

[3] http://clic.cimec.unitn.it/composes/semantic-vectors.html

[4] http://clic.cimec.unitn.it/dm

[5] A reviewer notes that composition rules could also be induced directly on the entailment task. This is an interesting possibility, but note that it would probably require a larger training set than we have available. Moreover, from a theoretical per-

sum the vectors of the words they contain to obtain sentence representations (Mitchell and Lapata, 2010). This approach, however, does not take into account word order, which is of obvious relevance to determining entailment between phrases. For example, *a dog chases a cat* does not entail *a cat chases a dog*, whereas each sentence entails itself. Therefore, we also used sentence vectors derived with the linguistically-motivated "practical lexical function" model (**plf**), that takes syntactic structure and word order into account (Paperno et al., 2014). In short, words acting as argument-taking functions (such as verbs) are not only associated to vectors, but also to one matrix for each argument they take (e.g., each transitive verb comes with a subject and an object matrix). Vector representations of arguments are recursively multiplied by function matrices, following the syntactic structure of a sentence. The final sentence representation is obtained by summing all the resulting vectors. We used pre-trained vector and matrix representations provided by Paperno and colleagues. Their setup is very comparable to the one of our count vectors: same source corpus, similar window size (3-word-window), positive PMI, and SVD reduction to 300 dimensions. The only notable differences are a vocabulary cut-off to the top 30K most frequent words in the corpus, and the use of content words only as windows.

4.2 Alternative entailment measures

As reviewed in Section 2, the literature on entailment with distributional methods has been dominated by the idea of feature inclusion. We thus compare BDSM to a variety of state-of-the art asymmetric similarity measures based on the distributional inclusion hypothesis (the dimensions of hyponym/antecedent vectors are included in those of their hypernyms/consequents). We consider the measures described in Lenci and Benotto (2012) (**clarkeDE**, **weedsPrec**, **cosWeeds**, and **invCL**), as well as **balAPinc**, which was shown to achieve optimal performance by Kotlerman et al. (2010). All these measures provide a score that is higher when a significant part of the candidate antecedent features (=dimensions) are included in those of the consequent. The measures are only meaningful when computed on a non-negative sparse space. Therefore, we evaluate them using the full count space. As an example, weedsPrec is computed as follows:

$$weedsPrec(u, v) = \frac{\sum_i \mathbb{1}[v_i > 0] \cdot u_i}{\sum_i u_i}$$

where u is the distributional vector of the antecedent, v that of the consequent.[6]

Finally, we implement a full-fledged supervised machine learning approach directly operating on distributional representations. Following the recent literature reviewed in Section 2 above, we train a Support Vector Machine (**SVM**) (Cristianini and Shawe-Taylor, 2000) on the concatenated distributional vectors of the training pairs, and judge the presence of entailment for a test pair based on the same concatenated representation (the results of Weeds et al. (2014) and Roller et al. (2014) suggest that concatenation is the most reliable way to construct SVM input representations that take both the antecedent and the consequent into account).

4.3 Data sets

Lexical entailment We test the models on benchmarks derived from two existing resources. We used the Lexical Entailment Data Set (LEDS) from Baroni et al. (2012) that contains both entailing (obtained by extracting hyponym-hypernym links from WordNet) and non-entailing pairs of words (constructed by reversing a third of the pairs and randomly shuffling the rest). We edited this resource by removing dubious data from the entailing pairs (e.g., *logo/signal*, *mankind/mammal*, *geek/performer*) and adding more negative cases (non-entailing pairs), obtained by shuffling words in the positive examples. We derived two balanced subsets: a development set (**LEDS-dev**) with 236 pairs in each class and a core set with 911 pairs in each class (**LEDS-core**), such that there is no lexical overlap between the positive classes of each set, and negative class overlap is minimized. Since a fair amount of negative cases were obtained by randomly shuffling words from the positive examples, leading to many unrelated couples, just pair similarity might be a

spective, we are interested in testing general methods of composition that are also good for other tasks (e.g., modeling sentence similarity), rather than developing *ad-hoc* composition rules specifically for entailment.

[6] BI is equivalent to weedsPrec in Boolean space.

	Positive	*Negative*
LEDS	elephant → animal	ape ↛ book
LEDS-dir		animal ↛ elephant
BLESS-coord	elephant → herbivore	elephant ↛ hippo
BLESS-mero		elephant ↛ trunk

Table 1: Lexical entailment examples.

Positive	*Negative*
A man is slowly trekking in the woods → The man is hiking in the woods	A group of scouts are camping in the grass ↛ A group of scouts are hiking through the grass

Table 2: SICK sentence entailment examples.

very strong baseline here. We thus explore a more challenging setup, **LEDS-dir**, where we replace the negative examples of LEDS-core by positive pairs in reverse order, thus focusing on entailment *direction*.

We derive two more benchmarks from BLESS (Baroni and Lenci, 2011). BLESS lists pairs of concepts linked by one of 5 possible relations: coordinates, hypernymy, meronymy, attributes and events. We employed this resource to construct **BLESS-coord**, which –unlike LEDS, where entailing pairs have to be distinguished from pairs of words that, mostly, bear no relation– is composed of 1,236 super-subordinate pairs (which we treat as positive examples) to be distinguished from 3,526 coordinate pairs. **BLESS-mero** has the same positive examples, but 2,943 holo-meronyms pairs as negatives. Examples of all lexical benchmarks are given in Table 1.

Sentence entailment To evaluate the models on recognizing entailment between sentences, we use a benchmark derived from SICK (Marelli et al., 2014b). The original data set contains pairs of sentences in entailment, contradiction and neutral relations. We focus on recognizing entailment, treating both contradictory and neutral pairs as negative examples (as in the classic RTE shared tasks up to 2008).[7] Data are divided into a development set (**SICK-dev**) with 500 sentence pairs (144 positive, 356 negative), a training set (**SICK-train**) with 4,500 pairs (1,299 positive, 3,201 negative) and a test set (**SICK-test**) with 4,927 pairs (1,414 positive, 3,513 negative). Examples from SICK are given in Table 2.

[7]This prevents a direct comparison with the results of the SICK shared task at SemEval (Marelli et al., 2014a). However, all competitive SemEval systems were highly engineered for the task, and made extensive use of a variety of pre-processing tools, features and external resources (cf. Table 8 of Marelli et al. (2014a)), so that a fair comparison with our simpler methods would not be possible in any case.

4.4 Training regime

We tune once and for all the hyperparameters of the models by maximizing accuracy on the small LEDS-dev set. For SVM, we tune the kernel type, picking a 2nd degree polynomial kernel for the count and TypeDM spaces, and a linear one for the predict space (alternatives: RBF and 1st, 2nd or 3rd degree polynomials). The choice for the count space is consistent with Turney and Mohammad (2014). For BDSM, we tune H (dimensionality of Boolean vectors), setting it to 100 for count, 1,000 for predict and 500 for TypeDM (alternatives: 10, 100, 500, 1,000 and 1,500) and the sparsity parameter ρ, picking 0.5 for count, 0.75 for predict, and 0.25 for TypeDM (alternatives: 0.01, 0.05, 0.1, 0.25, 0.5, 0.75). For BDSM and the asymmetric similarity measures, we also tune the e threshold above which a pair is treated as entailing for each dataset.

The γ (RBF kernel radius) and C (margin slackness) parameters of SVM and the λ, β and t parameters of BDSM (see Section 3) are set by maximizing accuracy on LEDS-dev for all lexical entailment experiments. For sentence entailment, we tune the same parameters on SICK-dev. In this case, given the imbalance between positive and negative pairs, we maximize weighted accuracy (that is, we count each true negative as $(|pos| + |neg|)/2|neg|$, and each true positive as $(|pos| + |neg|)/2|pos|$, where $|class|$ is the cardinality of the relevant class in the tuning data).

Finally, for lexical entailment, we train the SVM and BDSM weights by maximizing accuracy on LEDS-core. For LEDS-core and LEDS-dir evaluation, we use 10-fold validation. When evaluating on the BLESS benchmarks, we train on full LEDS-core, excluding any pairs also present in BLESS. For sentential entailment, the models are trained by

model	LEDS		BLESS	
	core	*dir*	*coord*	*mero*
count				
clarkeDE	77	63	27	36
weedsPrec	79	75	27	33
cosWeeds	79	63	26	35
invCL	77	63	27	36
balAPinc	79	66	26	36
SVM (count)	**84**	**90**	55	57
BDSM (count)	83	87	53	55
predict				
SVM (predict)	71	85	70	55
BDSM (predict)	80	79	**76**	**68**
TypeDM				
SVM (TypeDM)	78	83	56	60
BDSM (TypeDM)	83	71	31	59

Table 3: Percentage accuracy (LEDS) and F1 (BLESS) on the lexical entailment benchmarks.

maximizing weighted accuracy on SICK-train.

5 Lexical entailment

Table 3 reports lexical entailment results (percentage accuracies for the LEDS benchmarks, F1 scores for the unbalanced BLESS sets). We observe, first of all, that SVM and BDSM are clearly outperforming the asymmetric similarity measures in all tasks. In only one case the lowest performance attained by a supervised model drops below the level of the best asymmetric measure performance (BDSM using TypeDM on LEDS-dir).[8] The performance of the unsupervised measures, which rely most directly on the original distributional space, confirms that the latter is more suited to capture similarity than entailment. This is shown by the drop in performance from LEDS-core (where many negative examples are semantically unrelated) to LEDS-dir (where items in positive and negative pairs are equally similar), as well as by the increase from BLESS-coord to BLESS-mero (as coordinate negative examples are more tightly related than holomeronym pairs).

In the count input space, SVM and BDSM perform similarly across all 4 tasks, with SVM having a small edge. In the next sections, we will thus focus on count vectors, for the fairest comparison between the two models. BDSM reaches the most consistent results with predict vectors, where it performs particularly well on BLESS, and not dramatically worse than with count vectors on LEDS. On the other hand, predict vectors have a negative overall impact on SVM in 3 over 4 tasks. Concerning the interaction of input representations and tasks, we observe that count vectors work best with LEDS, whereas for BLESS predict vectors are the best choice, regardless of the supervised method employed.

Confirming the results of Baroni et al. (2014b), the TypeDM vectors are not a particularly good choice for either model. BDSM is specifically negatively affected by this choice in the LEDS-dir and BLESS-coord tasks. The tight taxonomic information captured by a dependency-based model such as TypeDM might actually be detrimental in tasks that require distinguishing between closely related forms, such as coordinates and hypernyms in BLESS-coord.

In terms of relative performance of the supervised entailment models, if one was to weigh each task equally, the best average performance would be reached by BDSM trained on predict vectors, with an average score of 75.75, followed by SVM on count vectors, with an average score of 71.5. We assess the significance of the difference between supervised models trained on the input vectors that give the best performance for each task by means paired t-tests on LEDS and McNemar tests on BLESS. SVM with count vectors is better than BDSM on LEDS-core (*not significant*) and LEDS-dir ($p<0.05$). On the other hand, BDSM with predict vectors is better than SVM on BLESS-coord ($p<0.001$) and BLESS-mero ($p<0.001$). We conclude that, overall, the two models perform similarly on lexical entailment tasks.

5.1 Learning efficiency

We just observed that SVM and BDSM have similar lexical entailment performance, especially in count space. However, the two models are radically differ-

[8] We also inspected ROC curves for BDSM (count) and the asymmetric measures, to check that the better performance of BDSM was not due to a brittle e (entailment threshold). The curves confirmed that, for all tasks, BDSM is clearly dominating all asymmetric measures across the whole e range.

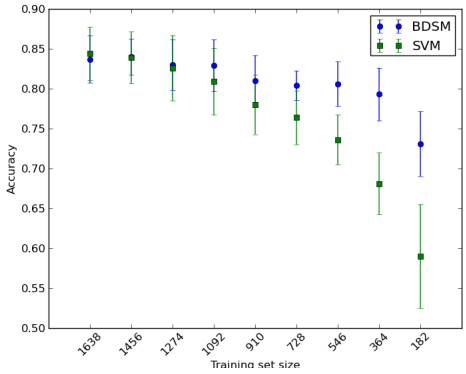

Figure 3: Average LEDS-core accuracy using count vectors in function of training set size.

ent in their structure. SVM fits a 2nd order polynomial separating entailing from non-entailing pairs in a space formed by the concatenation of their distributional representations. BDSM, on the other hand, finds a linear transformation into a space where features of the antecedent are included in those of the consequent. We conjecture that the latter has much larger *bias*, imposed by this strict subsective constraint.[9] We expect this bias to help learning, by limiting the search space and allowing the algorithm to harness training data in a more efficient way. Thus, BDSM should be better at learning with less data, where SVM will be prone to overfitting. To test this claim, we measured the cross-validated LEDS-core accuracy obtained from using vectors in count space when reducing the training items in steps of 182 pairs. The results can be seen in Figure 3. As expected, BDSM scales down much more gracefully, with accuracy well above 70% with as little as 182 training pairs.

6 Sentence entailment

Having shown in the previous experiments that the asymmetric measures are not competitive, we focus here on SVM and BDSM. As mentioned above in Section 5, we use count vectors for a fair comparison between the two models, based on their similar performance on the lexical benchmarks.

Recall that for sentence entailment we use the same hyperparameters as for the lexical tasks, that the model constants were tuned on SICK-dev, and the model weights on SICK-train (details in Section 4.4 above). Sentence representations are derived either with the plf approach, that returns sentence vectors built according to syntactic structure, or the additive (add) method, where constituent word vectors are simply summed to derive a sentence vector (see Section 4.1 above).

We compare SVM and BDSM to the **Sycophantic** baseline classifying all pairs as entailing and to a **Majority** baseline classifying everything as non-entailing. The Word Overlap method (**WO**) calculates the number of words in common between two sentences and classifies them as entailing whenever the ratio is above a certain threshold (calibrated on SICK-train).

Results are given in Table 4. Because of class unbalance, F1 is more informative than accuracy (the Majority baseline reaches the best accuracy with 0 precision and recall), so we focus on the former for analysis. We observe first that sentence vectors obtained with the additive model are consistently outperforming the more sophisticated plf approach. This confirms the results of Blacoe and Lapata (2012) on the effectiveness of simple composition methods. We leave it to further studies to determine to what extent this can be attributed to specific characteristics of SICK that make word order information redundant, and to what extent it indicates that plf is not exploiting syntactic information adequately (note that Paperno et al. (2014) report minimal performance differences between additive and plf for their *msrvid* benchmark, that is the closest to SICK).

Coming now to the crucial comparison of BDSM against SVM (focusing on the results obtained with the additive method), BDSM emerges as the best classifier when evaluated alone, improving over SVM, although the difference is not significant. Since the Word Overlap method is performing quite well (better than SVM) and the surface information used by WO should be complementary to the semantic cues exploited by the vector-based models, we built combined classifiers by training SVMs (on SICK-dev) with linear kernels and WO value plus each method's score (BI for BDSM and distance to the margin for SVM) as features. The combi-

[9] Mitchell (1980) defines bias as any basis for choosing one generalization over another, other than strict consistency with the observed training instances.

model	P	R	F1	A
Sycophantic	29	100	45	29
Majority	0	0	0	**71**
WO	40	86	55	60
SVM (add)	47	54	51	70
BDSM (add)	**48**	74	58	69
SVM (plf)	39	45	42	64
BDSM (plf)	44	71	55	66
SVM(add) + WO	44	**82**	58	65
BDSM(add) + WO	**48**	80	**60**	69
SVM(plf) + WO	42	76	54	63
BDSM(plf) + WO	42	77	54	63

Table 4: SICK results (percentages).

nations improve performance for both models and BDSM+WO attains the best overall F1 score, being statistically superior to both SVM+WO ($p<0.001$) and WO alone ($p < 0.001$) (statistical significance values obtained through McNemar tests).

We repeated the training data reduction experiment from Section 5.1 by measuring cross-validated F1 scores for SICK (with additive composition). We confirmed that BDSM is robust to decreasing the amount of training data, maintaining an F1 score of 56 with only 942 training items, whereas, with the same amount of training data, SVM drops to a F1 of 42.

7 Understanding Boolean vectors

BDSM produces representations that are meant to respect inclusion and be interpretable. We turn now to an extended analysis of the learned representations (focusing on those derived from count vectors), showing first how BDSM activation correlates with generality and abstractness, and then how similarity in BDSM space points in the direction of an extensional interpretation of Boolean units.

7.1 Boolean dimensions and generality

The BDSM layer is trained to assign more activation to a hypernym than its hyponyms (the hypernym units should include the hyponyms' ones), so the more general (that is, higher on the hypernymy scale) a concept is, the higher the proportion of activated units in its BDSM vector. The words that activate all nodes should be implied by all other terms. Indeed, very general words such as *thing(s)*, *everything*, and *anything* have Boolean vectors with all 1s. But there are also other words (a total of 768) mapping to the top element of the Boolean algebra (a vector of all 1s), including *reduction*, *excluded*, *results*, *benefit*, *global*, *extent*, *achieve*. The collapsing of these latter terms must be due to a combination of two factors: low dimensionality of Boolean space,[10] and the fact that the model was trained on a limited vocabulary, mostly consisting of concrete nouns, so there was simply no training evidence to characterize abstract words such as *benefit* in a more nuanced way.

Still, we predict that the proportion of Boolean dimensions that a word activates (i.e., dimensions with value 1) should correspond, as a trend, to its degree of semantic generality. More general concepts also tend to be more abstract, so we also expect a correlation between Boolean activation and the word rating on the concrete-abstract scale.[11] To evaluate these claims quantitatively, we rely on WordNet (Fellbaum, 1998), which provides an `is-a` hierarchy of word senses ('synsets') that can be used to measure semantic generality. We compute the average length of a path from the root of the hierarchy to the WordNet synsets of a word (shortest is most general, so that a higher depth score corresponds to a more *specific* concept). We further use the Ghent database (Brysbaert et al., 2013), that contains 40K English words rated on a 1-5 scale from least to most concrete (as expected, depth and concreteness are correlated, $\rho = .54$).

Boolean vector activation significantly correlates with both variables (ρ=-18 with depth, ρ=-30 with concreteness; these and all correlations below significant at $p < 0.005$). Moreover, the BDSM activations are much higher than those achieved by distributional vector L1 norm (which, surprisingly, has positive correlations: ρ=13 with depth, ρ=21 with concreteness) and word frequency (ρ=-2 with depth, ρ=4 with concreteness).

We visualize how Boolean activation correlates with generality in Figure 4. We plot the two example words *car* and *newspaper* together with their 30 nearest nominal neighbours in distributional

[10] With count input representations, our tuning favoured relatively dense 100-dimensional vectors (see Section 4.4).

[11] Automatically determining the degree of abstractness of concepts is a lively topic of research (Kiela et al., 2014; Turney et al., 2011).

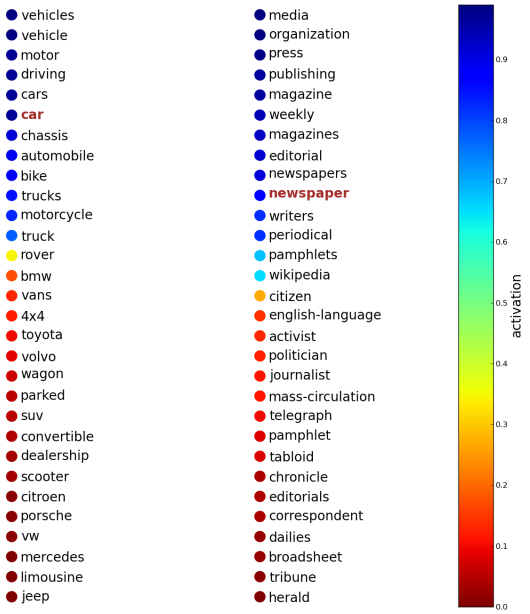

Figure 4: Boolean activation (percentage of positive dimensions) of the 30 nearest distributional neighbours of *car* and *newspaper*.

space,[12] sorting them from most to least activated. More general words do indeed cluster towards the top, while more specific words are pushed to the bottom. Interestingly, while *vehicle* and *organization* were present in the training data, that was not the case for *media* or *press*. Moreover, the training data did not contain any specific type of car (like *volvo* or *suv*) or newspaper (*tribune* or *tabloid*).

7.2 Similarity in Boolean space

From the model-theoretical point of view, word-to-BDSM mapping provides an *interpretation function* in the logical sense, mapping linguistic expressions to elements of the model domain (Boolean dimensions). If distributional vectors relate to concepts in a (hyper)intensional construal (Erk, 2013), Boolean vectors could encode their (possible) extensions along the lines suggested by Grefenstette (2013), with vector dimensions corresponding to entities in the domain of discourse.[13] Under the extensional interpretation, the Boolean vector of a word encodes the set of objects in the word extension. But of course, given that our BDSM implementation operates with only 100 dimensions, one cannot expect such an extensional interpretation of the model to be realistic. Still, the extensional interpretation of the Boolean model, while being highly idealized, makes some testable predictions. Under this view, synonyms should have identical Boolean vectors, antonyms should have disjoint vectors. Compatible terms (including hyponym-hypernym pairs) should overlap in their 1s. Cohyponyms, while high on the relatedness scale, should have low "extensional similarity"; *singer* and *drummer* are very related notions but the intersection of their extensions is small, and that between *alligator* and *crocodile* is empty (in real life, no entity is simultaneously a crocodile and an alligator).

As expected, the straightforward interpretation of dimensions as individuals in a possible world close to ours is contradicted by many counterexamples in the present BDSM implementation. For example, the nouns *man* and *woman* have a considerable overlap in activated Boolean dimensions, while in any plausible world hermaphrodite humans are rare. Still, compared to distributional space, BDSM goes in the direction of an extensional model as discussed above. To quantify this difference, we compared the similarity scores (cosines) produced by the two models. Specifically, we first created a list of pairs of semantically related words using the following procedure. We took the 10K most frequent words paired with their 10 closest neighbors in the count distributional space. We then filtered them to be of "medium frequency" (both words must lie within the 60K-90K frequency range in our 2.8B token corpus). One of the authors annotated the resulting 624 pairs as belonging to one of the following types: cohyponyms (137, e.g., *AIDS* vs. *diabetes*); derivationally related words (10, e.g., *depend* vs. *dependent*); hypernym-hyponym pairs (37, e.g., *arena* vs. *theater*); personal names (97, e.g., *Adams* vs. *Harris*); synonyms (including contextual ones; 49, e.g., *abilities* vs. *skill*); or "other" (294, e.g., *actress* vs. *starring*), if the pair does not fit any of the above types

[12] Due to tagging errors, the neighbors also include some verbs like *parked* or adjectives like *weekly*.

[13] In fact everything we say here applies equally well to certain *intensional* interpretations of Boolean vectors. For example, the atoms of the Boolean algebra could correspond not to entities in the actual world but to classes of individuals across possible worlds. Alternatively, one can think of the atoms as "typical cases" rather than actual individuals, or even as typical properties of the relevant individuals.

(some relations of interest, such as antonymy, were excluded from further analysis as they were instantiated by very few pairs). Since cosines have different distributions in distributional (**DS**) and Boolean space (**BS**), we z-normalized them before comparing those of pairs of the same type across the two spaces.

Under the extensional interpretation, we expect co-hyponyms to go apart after Boolean mapping, as they should in general have little extensional overlap. Indeed they have significantly lower cosines in BS than DS ($p < 0.001$; paired t-test). As expected under the extensional interpretation, personal names are very significantly less similar in BS than DS ($p < 0.001$). Synonyms and hypo/hypernyms have significant denotational overlap, and they move closer to each other after mapping. Specifically, synonyms significantly gain in similarity between BS and DS ($p<0.01$), whereas hyponym-hypernym pairs, while not differing significantly in average similarity across the spaces, change from being weakly significantly lower in cosine than all other pairs in DS ($p < 0.05$) to being indistinguishable from the other pairs in BS. Derivationally related words gain in similarity ($p<0.01$) collapsing to almost identical vectors after Boolean mapping. This deserves a special comment. Although words in these pairs typically belong to different parts of speech and are not synonyms in the usual sense, one could interpret them as denotational synonyms in the sense that they get reference in the same situations. Taking two word pairs from our data as examples, the existence of *experiments* entails the presence of something *experimental*, anything *Islamic* entails the presence of *Islam* in the situation, etc. If so, the fact that derivationally related words collapse under Boolean mapping makes perfect sense from the viewpoint of denotational overlap.

8 Conclusion

We introduced BDSM, a method that extracts representations encoding semantic properties relevant for making inferences from distributional semantic vectors. When applied to the task of detecting entailment between words or sentences, BDSM is competitive against a state-of-the-art SVM classifier, and needs less learning data to generalize. In contrast to SVM, BDSM is transparent: we are able not only to classify a pair of words (or sentences) with respect to entailment, but we also produce a compact Boolean vector for each word, that can be used alone for recognizing its entailment relations. Besides the analogy with the structures postulated in formal semantics, this can be important for practical applications that involve entailment recognition, where Boolean vectors can reduce memory and computing power requirements.

The Boolean vectors also allow for a certain degree of interpretability, with the number of active dimensions correlating with semantic generality and abstractness. Qualitative analysis suggests that Boolean mapping moves the semantic space from one organized around word relatedness towards a different criterion, where vectors of two words are closer to each other whenever their denotations have greater overlap. This is, however, just a tendency. Ideally, the overlap between dimensions of two vectors should be a measure of compatibility of concepts. In future research, we would like to explore to what extent one can reach this ideal, explicitly teaching the network to also capture other types of relations (e.g., no overlap between cohyponym representations), and using alternative learning methods.

We also want to look within the same framework at other phenomena, such as negation and conjunction, that have an elegant treatment in formal semantics but are currently largely outside the scope of distributional approaches.

Acknowledgements

We thank Yoav Goldberg, Omer Levy, Ivan Titov, Tim Rocktäschel, the members of the COMPOSES team (especially Angeliki Lazaridou) and the FLOSS reading group. We also thank Alexander Koller and the anonymous reviewers for their insightful comments. We acknowledge ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES)

References

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the EMNLP GEMS Workshop*, pages 1–10, Edinburgh, UK.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-Chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of EACL*, pages 23–32, Avignon, France.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014a. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9(6):5–110.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014b. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247, Baltimore, MD.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. In *Proceedings of *SEM*, pages 11–21, Atlanta, GA.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of AISTATS*, pages 127–135, La Palma, Canary Islands.

Samuel R. Bowman. 2013. Can recursive neural tensor networks learn logical reasoning? *CoRR*, abs/1312.6192.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, pages 1–8.

Gennaro Chierchia and Sally McConnell-Ginet. 2000. *Meaning and Grammar: An Introduction to Semantics*. MIT Press, Cambridge, MA.

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

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

Daoud Clarke. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics*, 38(1):41–71.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Nello Cristianini and John Shawe-Taylor. 2000. *An introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, Cambridge, UK.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: rationale, evaluation and approaches. *Natural Language Engineering*, 15:459–476.

Katrin Erk. 2009. Supporting inferences in semantic space: Representing words as regions. In *Proceedings of IWCS*, pages 104–115, Tilburg, Netherlands.

Katrin Erk. 2013. Towards a semantics for distributional representations. In *Proceedings of IWCS*, pages 95–106, Potsdam, Germany. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Dan Garrette, Katrin Erk, and Ray Mooney. 2013. A formal approach to linking logical form and vector-space lexical semantics. In H. Bunt, J. Bos, and S. Pulman, editors, *Computing Meaning, Vol. 4*, pages 27–48. Springer, Berlin.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of ACL*, pages 107–114, Ann Arbor, MI.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *Proceedings of *SEM*, pages 1–10.

Rodolphe Jenatton, Nicolas Le Roux, Antoine Bordes, and Guillaume Obozinski. 2012. A latent factor model for highly multi-relational data. In *Proceedings of NIPS*, pages 3176–3184, La Palma, Canary Islands.

Douwe Kiela, Felix Hill, Anna Korhonen, and Stephen Clark. 2014. Improving multi-modal representations using image dispersion: Why less is sometimes more. In *Proceedings of ACL*, pages 835–841, Baltimore, MD.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Thomas Landauer 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.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of *SEM*, pages 75–79, Montreal, Canada.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014a. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of SemEval*, pages 1–8, Dublin, Ireland.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella bernardi, and Roberto Zamparelli. 2014b. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*, pages 216–223, Rekjavik, Iceland.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. `http://arxiv.org/abs/1301.3781/`.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Tom M Mitchell. 1980. The need for biases in learning generalizations. Technical report, New Brunswick, NJ.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*, pages 90–99, Baltimore, MD.

Tamara Polajnar, Luana Fagarasan, and Stephen Clark. 2014. Reducing dimensions of tensors in type-driven distributional semantics. In *Proceedings of EMNLP*, pages 1036–1046, Doha, Qatar.

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of EACL*, pages 511–519, Gothenburg, Sweden.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of HLT-NAACL*, pages 1119–1129, Denver, CO.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING*, pages 1025–1036, Dublin, Ireland.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of EACL*, pages 38–42, Gothenburg, Sweden.

T. Sasao. 1999. *Switching Theory for Logic Synthesis*. Springer, London.

Hinrich Schütze. 1997. *Ambiguity Resolution in Natural Language Learning*. CSLI, Stanford, CA.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, pages 926–934, Lake Tahoe, NV.

Peter Turney and Said Mohammad. 2014. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*. In press.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of EMNLP*, pages 680–690, Edinburgh, UK.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (Linear) maps of the impossible: Capturing semantic anomalies in distributional space. In *Proceedings of the ACL Workshop on Distributional Semantics and Compositionality*, pages 1–9, Portland, OR.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of COLING*, pages 1015–1021, Geneva, Switzerland.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING*, pages 2249–2259, Dublin, Ireland.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.