Integrating Multiplicative Features into Supervised Distributional Methods for Lexical Entailment

Tu Vu
College of Information and Computer Sciences
University of Massachusetts Amherst
Amherst, MA, USA
tuvu@cs.umass.edu

Vered Shwartz
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com

Abstract

Supervised distributional methods are applied successfully in lexical entailment, but recent work questioned whether these methods actually learn a relation between two words. Specifically, Levy et al. (2015) claimed that linear classifiers learn only separate properties of each word. We suggest a cheap and easy way to boost the performance of these methods by integrating multiplicative features into commonly used representations. We provide an extensive evaluation with different classifiers and evaluation setups, and suggest a suitable evaluation setup for the task, eliminating biases existing in previous ones.

1 Introduction

Lexical entailment is concerned with identifying the semantic relation, if any, holding between two words, as in *(pigeon, hyponym, animal)*. The popularity of the task stems from its potential relevance to various NLP applications, such as question answering and recognizing textual entailment (Dagan et al., 2013) that often rely on lexical semantic resources with limited coverage like Wordnet (Miller, 1995). Relation classifiers can be used either within applications or as an intermediate step in the construction of lexical resources which is often expensive and time-consuming.

Most methods for lexical entailment are distributional, i.e., the semantic relation holding between x and y is recognized based on their distributional vector representations. While the first methods were unsupervised and used high-dimensional sparse vectors (Weeds and Weir, 2003; Kotlerman et al., 2010; Santus et al., 2014), in recent years, supervised methods became popular (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). These methods are mostly based on word embeddings (Mikolov et al., 2013b; Pennington et al., 2014a) utilizing various vector combinations that are designed to capture relational information between two words.

While most previous work reported success using supervised methods, some questions remain unanswered: First, several works suggested that supervised distributional methods are incapable of inferring the relationship between two words, but rather rely on independent properties of each word (Levy et al., 2015; Roller and Erk, 2016; Shwartz et al., 2016), making them sensitive to training data; Second, it remains unclear what is the most appropriate representation and classifier; previous studies reported inconsistent results with **Concat**$\langle \vec{v_x} \oplus \vec{v_y} \rangle$ (Baroni et al., 2012) and **Diff**$\langle \vec{v_y} - \vec{v_x} \rangle$ (Roller et al., 2014; Weeds et al., 2014; Fu et al., 2014), using various classifiers.

In this paper, we investigate the effectiveness of multiplicative features, namely, the element-wise multiplication **Mult**$\langle \vec{v_x} \odot \vec{v_y} \rangle$, and the squared difference **Sqdiff**$\langle (\vec{v_y} - \vec{v_x}) \odot (\vec{v_y} - \vec{v_x}) \rangle$. These features, similar to the cosine similarity and the Euclidean distance, might capture a different notion of interaction information about the relationship holding between two words. We directly integrate them into some commonly used representations. For instance, we consider the concatenation **Diff**$\oplus$**Mult** $\langle (\vec{v_y} - \vec{v_x}) \oplus (\vec{v_x} \odot \vec{v_y}) \rangle$ that might capture both the typicality of each word in the relation (e.g., if y is a typical hypernym) and the similarity between the words.

We experiment with multiple supervised distributional methods and analyze which representations perform well in various evaluation setups. Our analysis confirms that integrating multiplicative features into standard representations can substantially boost the performance of linear classifiers. While the contribution over non-linear classifiers is sometimes marginal, they are expensive to train, and linear classifiers can achieve the same effect "cheaply" by integrating multiplicative fea-

*Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 160–166
New Orleans, June 5-6, 2018.

tures. The contribution of multiplicative features is mostly prominent in strict evaluation settings, i.e., lexical split (Levy et al., 2015) and out-of-domain evaluation that disable the models' ability to achieve good performance by memorizing words seen during training. We find that **Concat** $\oplus$ **Mult** performs consistently well, and suggest it as a strong baseline for future research.

2 Related Work

Available Representations In supervised distributional methods, a pair of words (x, y) is represented as some combination of the word embeddings of x and y, most commonly **Concat** $\langle \vec{v_x} \oplus \vec{v_y} \rangle$ (Baroni et al., 2012) or **Diff** $\langle \vec{v_y} - \vec{v_x} \rangle$ (Weeds et al., 2014; Fu et al., 2014).

Limitations Recent work questioned whether supervised distributional methods actually learn the relation between x and y or only separate properties of each word. Levy et al. (2015) claimed that they tend to perform "lexical memorization", i.e., memorizing that some words are prototypical to certain relations (e.g., that $y =$ *animal* is a hypernym, regardless of x). Roller and Erk (2016) found that under certain conditions, these methods actively learn to infer hypernyms based on separate occurrences of x and y in Hearst patterns (Hearst, 1992). In either case, they only learn whether x and y independently match their corresponding slots in the relation, a limitation which makes them sensitive to the training data (Shwartz et al., 2017; Sanchez and Riedel, 2017).

Non-linearity Levy et al. (2015) claimed that the linear nature of most supervised methods limits their ability to capture the relation between words. They suggested that using support vector machine (SVM) with non-linear kernels slightly mitigates this issue, and proposed KSIM, a custom kernel with multiplicative integration.

Multiplicative Features The element-wise multiplication has been studied by Weeds et al. (2014), but models that operate exclusively on it were not competitive to **Concat** and **Diff** on most tasks. Roller et al. (2014) found that the squared difference, in combination with **Diff**, is useful for hypernymy detection. Nevertheless, little to no work has focused on investigating combinations of representations obtained by concatenating various base representations for the more general task of lexical entailment.

Base representations	Combinations
Only-x $\langle \vec{v_x} \rangle$	**Diff** $\oplus$ **Mult**
Only-y $\langle \vec{v_y} \rangle$	**Diff** $\oplus$ **Sqdiff**
Diff $\langle \vec{v_y} - \vec{v_x} \rangle$	**Sum** $\oplus$ **Mult**
Sum $\langle \vec{v_x} + \vec{v_y} \rangle$	**Sum** $\oplus$ **Sqdiff**
Concat $\langle \vec{v_x} \oplus \vec{v_y} \rangle$	**Concat** $\oplus$ **Mult**
Mult $\langle \vec{v_x} \odot \vec{v_y} \rangle$	**Concat** $\oplus$ **Sqdiff**
Sqdiff $\langle (\vec{v_y} - \vec{v_x}) \odot (\vec{v_y} - \vec{v_x}) \rangle$	

Table 1: Word pair representations.

3 Methodology

We classify each word pair (x, y) to a specific semantic relation that holds for them, from a set of pre-defined relations (i.e., multiclass classification), based on their distributional representations.

3.1 Word Pair Representations

Given a word pair (x, y) and their embeddings $\vec{v_x}, \vec{v_y}$, we consider various compositions as feature vectors for classifiers. Table 1 displays base representations and combination representations, achieved by concatenating two base representations.

3.2 Word Vectors

We used 300-dimensional pre-trained word embeddings, namely, GloVe (Pennington et al., 2014b) containing 1.9M word vectors trained on a corpus of web data from Common Crawl (42B tokens),[1] and Word2vec (Mikolov et al., 2013a,c) containing 3M word vectors trained on a part of Google News dataset (100B tokens).[2] Out-of-vocabulary words were initialized randomly.

3.3 Classifiers

Following previous work (Levy et al., 2015; Roller and Erk, 2016), we trained different types of classifiers for each word-pair representation outlined in Section 3.1, namely, logistic regression with L_2 regularization (LR), SVM with a linear kernel (LIN), and SVM with a Gaussian kernel (RBF). In addition, we trained multi-layer perceptrons with a single hidden layer (MLP). We compare our models against the KSIM model found to be successful in previous work (Levy et al., 2015; Kruszewski et al., 2015). We do not include Roller and Erk (2016)'s model since it focuses only on hypernymy. Hyper-parameters are tuned using grid search, and we report the test performance of the

[1] http://nlp.stanford.edu/projects/glove/
[2] http://code.google.com/p/word2vec/

Dataset	Relations	#Instances	#Domains
BLESS	attri (attribute), coord (co-hyponym), event, hyper (hypernymy), mero (meronymy), random	26,554	17
K&H+N	hypo (hypernymy), mero (meronymy), sibl (co-hyponym), false (random)	63,718	3
ROOT09	hyper (hypernymy), coord (co-hyponym), random	12,762	–
EVALution	HasProperty (attribute), synonym, HasA (possession), MadeOf (meronymy), IsA (hypernymy), antonym, PartOf (meronymy)	7,378	–

Table 2: Metadata on the datasets. Relations are mapped to corresponding WordNet relations, if available.

hyper-parameters that performed best on the validation set. Below are more details about the training procedure:

- For LR, the inverse of regularization strength is selected from $\{2^{-1}, 2^1, 2^3, 2^5\}$.
- For LIN, the penalty parameter C of the error term is selected from $\{2^{-5}, 2^{-3}, 2^{-1}, 2^1\}$.
- For RBF, C and γ values are selected from $\{2^1, 2^3, 2^5, 2^7\}$ and $\{2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}\}$, respectively.
- For MLP, the hidden layer size is either 50 or 100, and the learning rate is fixed at 10^{-3}. We use early stopping based on the performance on the validation set. The maximum number of training epochs is 100.
- For KSIM, C and α values are selected from $\{2^{-7}, 2^{-5}, \ldots, 2^7\}$ and $\{0.0, 0.1, \ldots, 1.0\}$, respectively.

3.4 Datasets

We evaluated the methods on four common semantic relation datasets: BLESS (Baroni and Lenci, 2011), K&H+N (Necsulescu et al., 2015), ROOT09 (Santus et al., 2016), and EVALution (Santus et al., 2015). Table 2 provides metadata on the datasets. Most datasets contain word pairs instantiating different, explicitly typed semantic relations, plus a number of unrelated word pairs (*random*). Instances in BLESS and K&H+N are divided into a number of topical domains.[3]

3.5 Evaluation Setup

We consider the following evaluation setups:

Random (RAND) We randomly split each dataset into 70% train, 5% validation and 25% test.

Lexical Split (LEX) In line with recent work (Shwartz et al., 2016), we split each dataset into train, validation and test sets so that each contains a distinct vocabulary. This differs from Levy et al. (2015) who dedicated a subset of the train set for evaluation, allowing the model to memorize when tuning hyper-parameters. We tried to keep the same ratio $70 : 5 : 25$ as in the random setup.

Out-of-domain (OOD) To test whether the methods capture a generic notion of each semantic relation, we test them on a domain that the classifiers have not seen during training. This setup is more realistic than the random and lexical split setups, in which the classifiers can benefit from memorizing verbatim words (random) or regions in the vector space (lexical split) that fit a specific slot of each relation.

Specifically, on BLESS and K&H+N, one domain is held out for testing whilst the classifiers are trained and validated on the remaining domains. This process is repeated using each domain as the test set, and each time, a randomly selected domain among the remaining domains is left out for validation. The average results are reported.

4 Experiments

Table 3 summarizes the best performing base representations and combinations on the test sets across the various datasets and evaluation setups.[4] The results across the datasets vary substantially in some cases due to the differences between the datasets' relations, class balance, and the source from which they were created. For instance, K&H+N is imbalanced between the number of instances across relations and domains. ROOT09 was designed to mitigate the lexical memorization issue by adding negative switched hyponym-hypernym pairs to the dataset, making it an inherently more difficult dataset. EVALution contains a richer set of semantic relations. Overall, the addition of multiplicative features improves upon the performance of the base representations.

Classifiers Multiplicative features substantially boost the performance of linear classifiers. However, the gain from adding multiplicative features

[3] We discarded two relations in EVALution with too few instances and did not include its domain information since each word pair can belong to multiple domains at once.

[4] Due to the space limitation, we only show the results obtained with Glove. The trend is similar across the word embeddings.

Setup	Dataset	Linear classifiers (LR, LIN)					Non-linear classifiers (RBF, MLP)					KSIM
		$\vec{v_y}$	Base		Combination		$\vec{v_y}$	Base		Combination		
RAND	BLESS	84.4	LR Concat	83.8	LR Concat ⊕ Mult	89.5 (**+5.7**)	89.3	RBF Concat	94.0	RBF Concat ⊕ Mult	94.3 (**+0.3**)	70.2
	K&H-N	89.1	LR Concat	95.4	LR Concat ⊕ SqDiff	96.1 (**+0.7**)	96.4	RBF Concat	98.6	RBF Concat ⊕ Mult	98.6 (**0.0**)	82.4
	ROOT09	68.5	LIN Sum	65.9	LIN Sum ⊕ Mult	84.6 (**+18.7**)	66.1	RBF Sum	87.3	RBF Sum ⊕ SqDiff	88.8 (**+1.5**)	72.3
	EVALution	49.7	LIN Concat	56.7	LIN Concat ⊕ Mult	56.8 (**+0.1**)	52.1	RBF Concat	61.1	RBF Concat ⊕ Mult	60.6 (**-0.5**)	50.5
LEX	BLESS	69.9	LIN Concat	70.6	LIN Concat ⊕ Mult	74.5 (**+3.9**)	69.8	MLP Concat	63.0	MLP Concat ⊕ Mult	73.8 (**+10.8**)	65.8
	K&H-N	78.3	LIN Sum	74.0	LIN Sum ⊕ SqDiff	76.1 (**+2.1**)	83.2	RBF Sum	82.0	RBF Sum ⊕ Mult	81.7 (**-0.3**)	77.5
	ROOT09	66.7	LR Concat	66.0	LR Concat ⊕ Mult	77.9 (**+11.9**)	64.5	RBF Concat	76.8	RBF Concat ⊕ Mult	81.6 (**+4.8**)	66.7
	EVALution	35.0	LR Concat	37.9	LR Concat ⊕ Mult	40.2 (**+2.3**)	35.5	RBF Concat	43.1	RBF Concat ⊕ Mult	44.9 (**+1.8**)	35.9
OOD	BLESS	70.9	LIN Concat	69.9	LIN Concat ⊕ Mult	77.0 (**+7.1**)	69.9	RBF Diff	78.7	RBF Diff ⊕ Mult	81.5 (**+2.8**)	57.8
	K&H-N	38.5	LIN Concat	38.6	LIN Concat ⊕ Mult	39.7 (**+1.1**)	48.6	MLP Sum	44.7	MLP Sum ⊕ Mult	47.9 (**+3.2**)	48.9

Table 3: Best test performance (F_1) across different datasets and evaluation setups, using Glove. The number in brackets indicates the performance gap between the best performing combination and base representation setups.

Vector/ Classifier		RAND							OOD						
		$\vec{v_y}$	Diff	Diff ⊕ Mult	Sum	Sum ⊕ Mult	Concat	Concat ⊕ Mult	$\vec{v_y}$	Diff	Diff ⊕ Mult	Sum	Sum ⊕ Mult	Concat	Concat ⊕ Mult
GloVe	LR	84.4	81.5	87.6 (**+6.1**)	81.5	87.0 (**+5.5**)	83.8	89.5 (**+5.7**)	70.9	64.5	74.7 (**+10.2**)	59.2	68.9 (**+9.7**)	69.5	76.5 (**+7.0**)
	LIN	84.1	81.5	87.7 (**+6.2**)	81.3	87.2 (**+5.9**)	83.8	89.2 (**+5.4**)	70.7	64.6	74.8 (**+10.2**)	59.3	69.4 (**+10.1**)	69.9	77.0 (**+7.1**)
	RBF	89.3	93.8	94.1 (**+0.3**)	94.4	94.2 (**-0.2**)	94.0	**94.3** (**+0.3**)	67.8	78.7	**81.5** (**+2.8**)	65.3	66.4 (**+1.1**)	69.5	75.7 (**+6.2**)
	MLP	84.4	87.4	89.2 (**+1.8**)	87.2	89.9 (**+2.7**)	90.5	90.5 (**0.0**)	69.9	67.4	77.7 (**+10.3**)	57.3	66.1 (**+8.8**)	71.5	77.3 (**+5.8**)
Word2vec	LR	83.5	81.0	85.4 (**+4.4**)	80.0	84.6 (**+4.6**)	83.6	87.1 (**+3.5**)	71.2	62.4	69.0 (**+6.6**)	59.0	65.3 (**+6.3**)	71.8	**76.1** (**+4.3**)
	LIN	83.3	80.8	84.6 (**+3.8**)	80.4	84.5 (**+4.1**)	83.3	86.5 (**+3.2**)	71.5	62.8	69.1 (**+6.3**)	59.8	65.2 (**+5.4**)	72.1	76.0 (**+3.9**)
	RBF	89.1	93.7	93.7 (**0.0**)	93.7	**93.8** (**+0.1**)	93.6	**93.8** (**+0.2**)	69.2	75.6	76.0 (**+0.4**)	64.7	66.3 (**+1.6**)	71.4	75.3 (**+3.9**)
	MLP	81.6	81.0	84.6 (**+3.6**)	79.6	85.2 (**+5.6**)	81.3	84.7 (**+3.4**)	70.2	63.4	69.3 (**+5.9**)	56.2	60.0 (**+3.8**)	70.5	74.6 (**+4.1**)

Table 4: Test performance (F_1) on BLESS in the RAND and OOD setups, using Glove and Word2vec.

is smaller when non-linear classifiers are used, since they partially capture such notion of interaction (Levy et al., 2015). Within the same representation, there is a clear preference to non-linear classifiers over linear classifiers.

Evaluation Setup The **Only-y** representation indicates how well a model can perform without considering the relation between x and y (Levy et al., 2015). Indeed, in RAND, this method performs similarly to the others, except on ROOT09, which by design disables lexical memorization. As expected, a general decrease in performance is observed in LEX and OOD, stemming from the methods' inability to benefit from lexical memorization. In these setups, there is a more significant gain from using multiplicative features when non-linear classifiers are used.

Word Pair Representations Among the base representations **Concat** often performed best, while **Mult** seemed to be the preferred multiplicative addition. **Concat** ⊕ **Mult** performed consistently well, intuitively because **Concat** captures the typicality of each word in the relation (e.g., if y is a typical hypernym) and **Mult** captures the similarity between the words (where **Concat** alone may suggest that *animal* is a hypernym of *apple*). To take a closer look at the gain from adding **Mult**, Table 4 shows the performance of the various base representations and combinations with **Mult** using different classifiers on BLESS.[5]

5 Analysis of Multiplicative Features

We focus the rest of the discussion on the OOD setup, as we believe it is the most challenging setup, forcing methods to consider the relation between x and y. We found that in this setup, all methods performed poorly on K&H+N, likely due to its imbalanced domain and relation distribution. Examining the per-relation F_1 scores, we see that many methods classify all pairs to one relation. Even KSIM, the best performing method in this

[5] We also tried $\vec{v_x}$ with multiplicative features but they performed worse.

x	**relation**	y	**similarity**	**Concat**	**Concat ⊕ Mult**
cloak-n	random	good-j	0.195	attribute	random
cloak-n	random	hurl-v	0.161	event	random
cloak-n	random	stop-v	0.186	event	random
coat-n	event	wear-v	0.544	random	event
cloak-n	mero	silk-n	0.381	random	mero
dress-n	attri	feminine-j	0.479	random	attri

Table 5: Example pairs which were incorrectly classified by **Concat** while being correctly classified by **Concat** ⊕ **Mult** in BLESS, along with their cosine similarity scores.

setup, classifies pairs as either *hyper* or *random*, effectively only determining if they are related or not. We therefore focus our analysis on BLESS.

To get a better intuition of the contribution of multiplicative features, Table 5 exemplifies pairs that were incorrectly classified by **Concat** (RBF) while correctly classified by **Concat** ⊕ **Mult** (RBF), along with their cosine similarity scores. It seems that **Mult** indeed captures the similarity between x and y. While **Concat** sometimes relies on properties of a single word, e.g. classifying an adjective y to the *attribute* relation and a verb y to the *event* relation, adding **Mult** changes the classification of such pairs with low similarity scores to *random*. Conversely, pairs with high similarity scores which were falsely classified as *random* by **Concat** are assigned specific relations by **Concat** ⊕ **Mult**.

Interestingly, we found that across domains, there is an almost consistent order of relations with respect to mean intra-pair cosine similarity:

coord	**meronym**	**attribute**	**event**	**hypernym**	**random**
0.426	0.323	0.304	0.296	0.279	0.141

Table 6: Mean pairwise cosine similarity in BLESS.

Since the difference between *random* (0.141) and other relations (0.279-0.426) was the most significant, it seems that multiplicative features help distinguishing between related and unrelated pairs. This similarity is possibly also used to distinguish between other relations.

6 Conclusion

We have suggested a cheap way to boost the performance of supervised distributional methods for lexical entailment by integrating multiplicative features into standard word-pair representations. Our results confirm that the multiplicative features boost the performance of linear classifiers, and in strict evaluation setups, also of non-linear classifiers. We performed an extensive evaluation with different classifiers and evaluation setups, and suggest the out-of-domain evaluation as the most suitable for the task. Directions for future work include investigating other compositions, and designing a neural model that can automatically learn such features.

7 Acknowledgements

We would like to thank Wei Lu for his involvement and advice in the early stage of this project, Stephen Roller and Omer Levy for valuable discussions, and the anonymous reviewers for their insightful comments and suggestions.

Vered is supported in part by an Intel ICRI-CI grant, the Israel Science Foundation grant 1951/17, the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), the Clore Scholars Programme (2017), and the AI2 Key Scientific Challenges Program (2017).

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France. Association for Computational Linguistics.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209, Baltimore, Maryland. Association for Computational Linguistics.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545.

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

Germn Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR, 2013*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A. Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014a. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014b. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2163–2172, Austin, Texas. Association for Computational Linguistics.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ivan Sanchez and Sebastian Riedel. 2017. How well can we predict hypernyms from word embeddings? a dataset-centric analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 401–407, Valencia, Spain. Association for Computational Linguistics.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Nine features in a random forest to learn taxonomical semantic relations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 38–42, Gothenburg, Sweden. Association for Computational Linguistics.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics (LDL-2015)*, pages 64–69.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398, Berlin, Germany. Association for Computational Linguistics.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 65–75, Valencia, Spain. Association for Computational Linguistics.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Julie Weeds and David Weir. 2003. *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, chapter A General Framework for Distributional Similarity.