A Latent Variable Model for Learning Distributional Relation Vectors

Jose Camacho-Collados[1] , **Luis Espinosa-Anke**[1] , **Shoaib Jameel**[2] and **Steven Schockaert**[1]
[1]School of Computer Science and Informatics, Cardiff University, United Kingdom
[2]School of Computing, University of Kent, United Kingdom
{camachocolladosj,espinosa-ankel,schockaerts1}@cardiff.ac.uk, m.s.jameel@kent.ac.uk

Abstract

Recently a number of unsupervised approaches have been proposed for learning vectors that capture the relationship between two words. Inspired by word embedding models, these approaches rely on co-occurrence statistics that are obtained from sentences in which the two target words appear. However, the number of such sentences is often quite small, and most of the words that occur in them are not relevant for characterizing the considered relationship. As a result, standard co-occurrence statistics typically lead to noisy relation vectors. To address this issue, we propose a latent variable model that aims to explicitly determine what words from the given sentences best characterize the relationship between the two target words. Relation vectors then correspond to the parameters of a simple unigram language model which is estimated from these words.

1 Introduction

Word embedding models, which learn vector representations of word meaning based on co-occurrence statistics, are now well-established [Mikolov *et al.*, 2013; Bojanowski *et al.*, 2017]. Recently, inspired by the success of these approaches, a number of methods have been proposed for learning vector representations of *relations* between words [Washio and Kato, 2018a; Jameel *et al.*, 2018; Espinosa-Anke and Schockaert, 2018; Washio and Kato, 2018b; Joshi *et al.*, 2019]. This idea of encoding relations as vectors stands in stark contrast with the traditional treatment of relations in Natural Language Processing (NLP), where the focus has mostly been on well-defined, discrete relation types.

The use of vector representations means that we can, in principle, capture aspects of meaning which go beyond what can be expressed using discrete representations, such as degrees of lexical entailment [Vulić *et al.*, 2017] or relational similarity [Jurgens *et al.*, 2012]. Relation vectors can also capture relation types that are not usually included in knowledge graphs. Consider, for instance, the relation between *dog* and *garden*. While intuitively clear (e.g. the fact that dogs enjoy spending time in the garden), such relations are missing from traditional knowledge bases. In contrast, consider the following sentences from Wikipedia[1]:

... pretty little dog that **led** her **to** a **lovely** garden. (1)

... his dog, Bendicò, **joyfully** dig up the garden, ... (2)

... for the dog, who **carried** it **into** the garden to eat. (3)

While such sentences do not explicitly assert the relationship between *dog* and *garden*, it can to a large extent be characterized based on the words that occur in them. For instance, to this end, the method used in Espinosa-Anke and Schockaert [2018] simply averages the pre-trained word vectors of the words that occur between *dog* and *garden* (while averages of the words occurring before *dog* and of the words occurring after *garden* are also used as additional context). Intuitively, however, simply averaging word vectors from sentences such as (1)–(3) might not be ideal, as only some words are relevant (i.e. the ones shown in bold). One possibility is to take into account how strongly each of the context words from these sentences is related to the considered word pair. For instance, Jameel *et al.* [2018] used generalizations of point-wise mutual information (PMI) to three arguments for this purpose. One difficulty with this strategy is that such statistics have to be computed from very limited amounts of information (e.g. a word such as *joyfully* may only occur once, even though it is clearly relevant). Furthermore, the most relevant words are often common words such as prepositions, which are less likely to be identified using PMI-like statistics.

To address these concerns, in this paper we introduce RELATIVE (RELations as LATent dIscourse VEctors), a new method for learning relation vectors based on the following intuition. Given a word pair (a, b) and a context word w, we can distinguish four possibilities: (i) w relates to the relation between a and b; (ii) w only relates to a; (iii) w only relates to b; (iv) w is neither related to a nor to b. Our method uses a simple unigram language model to estimate the probability of each of these four cases, for every word w that occurs in a sentence mentioning both a and b. Similarly to the aforementioned PMI-based weighting strategies, this approach then naturally leads us to compute relation vectors as a weighted average of word vectors. We empirically show that our relation vectors outperform those from existing methods.

[1]https://en.wikipedia.org/wiki/The_Story_of_the_Queen_of_the_Flowery_Isles, https://en.wikipedia.org/wiki/The_Leopard, https://en.wikipedia.org/wiki/Lump_(dog)

2 Related Work

The problem of relation extraction has already been widely studied in NLP, but the focus has traditionally been on extracting instances of a given well-defined relation (e.g. is-a, part-of, has-capital), for instance by hand-crafting linguistic patterns that are indicative of the considered relation [Hearst, 1992] or by learning such patterns from a set of seed examples [Agichtein and Gravano, 2000]. Some approaches have also been proposed which are not restricted to a pre-defined set of relation types. The NELL system [Carlson *et al.*, 2010] is perhaps the best-known example of this kind. While it starts off with a set of seed examples, it then aims to automatically extend the set of considered relation types with only minimal amounts of human supervision. Fully unsupervised approaches to relation extraction have also been proposed [Shinyama and Sekine, 2006; Banko *et al.*, 2007]. These methods essentially focus on identifying clusters of linguistic patterns that tend to link the same word pairs, and are therefore assumed to correspond to a semantically coherent relationship. These methods are somewhat closer in spirit to our approach, which is also fully unsupervised, but with the crucial difference that we use vector representations. This, among others, means that we can easily incorporate our representations into neural NLP architectures.

In recent years, the focus in relation extraction has shifted to the use of neural network models [dos Santos *et al.*, 2015; Xu *et al.*, 2015]. Different from our setting, such approaches are supervised and limited to a fixed set of relation types. Perhaps surprisingly, however, it is also possible to implement competitive relation extraction methods which rely on vector representations that are learned in an unsupervised way. For instance, Hashimoto *et al.* [2015] predict which relations are asserted in a given sentence by training a logistic regression classifier on a vector representation of the sentence, where the latter is obtained in an unsupervised way by averaging pre-trained word vectors. Along similar lines, Jameel *et al.* [2018] obtained state-of-the-art performance in a relation extraction task by training an SVM classifier on relation vectors that were obtained in an unsupervised way.

The problem of learning unsupervised relation vectors goes back at least to Turney [2005], who proposed a method based on singular value decomposition, starting from a matrix which encodes how often different word pairs are linked by different linguistic patterns. Along similar lines, Riedel *et al.* [2013] learned vector representations of word pairs, but they combined statistics about linguistic patterns with triples from a knowledge graph. More recently, Jameel *et al.* [2018] proposed an unsupervised method for learning relation vectors which is inspired by the GloVe word embedding model. Their training objective is to learn vector representations $\mathbf{r_{ab}}$ of word pairs and vector representations $\tilde{\mathbf{w}_c}$ of context words, such that the dot product $\mathbf{r_{ab}} \cdot \tilde{\mathbf{w}_c}$ predicts the strength of association between occurrences of the context word c and the word pair (a, b) in a sentence. For this purpose, they considered a number of generalizations of PMI to three arguments. A simpler and more efficient alternative was proposed in Espinosa-Anke and Schockaert [2018], where relation vectors were learned by averaging the word vectors of the context words appearing in sentences that contain the word pair (a, b) and then using a conditional autoencoder.

The aforementioned methods have the disadvantage that they can only learn relation vectors for pairs of words that co-occur in the same sentence sufficiently often. To address this, a number of methods have been proposed which learn word vectors that are aimed at modelling relational properties [Washio and Kato, 2018a; Washio and Kato, 2018b; Joshi *et al.*, 2019]. Specifically, these works train a neural network that maps the concatenation of two word vectors $\mathbf{w_a} \oplus \mathbf{w_b}$ to a vector $\mathbf{r_{ab}}$ which represents the relation between the two corresponding words a and b. This network is trained such that $\mathbf{r_{ab}}$ captures the contexts in which the word pair appears, where contexts correspond to learned vector encodings of dependency paths [Washio and Kato, 2018a] or LSTM-based neural encodings of surface patterns [Washio and Kato, 2018b; Joshi *et al.*, 2019].

3 Model Description

Our approach is inspired by the averaging based sentence embedding method from Arora *et al.* [2017], which essentially views sentence vectors as latent parameters of a simple unigram language model. In Section 3.1, we first briefly describe this method. Then, in Section 3.2 we introduce our model.

Notations. For a given word w, we will write $\mathbf{w}$ for the corresponding word vector from a given pre-trained word embedding. We write $\|\mathbf{w}\|$ for its Euclidean norm, and we define $\text{norm}(\mathbf{w}) = \frac{\mathbf{w}}{\|\mathbf{w}\|}$ for $\mathbf{w} \neq \mathbf{0}$.

3.1 Latent Discourse Vectors

In Arora *et al.* [2016] a generative unigram language model is studied in which the i^{th} word from a document is assumed to have been sampled as

$$w_i \propto \exp(\mathbf{w_i} \cdot \mathbf{c_i}) \tag{4}$$

where $\mathbf{c_i}$ is a latent discourse vector of unit norm. From this generative model, a simple approach for constructing sentence embeddings can be obtained by assuming that the discourse vector remains constant throughout each sentence $S = w_1....w_k$. In that case, the maximum likelihood estimate of the sentence-specific discourse vector is given by:

$$\hat{\mathbf{c_S}} = \text{norm}\left(\sum_{i=1}^{k} \mathbf{w_i}\right) \tag{5}$$

In Arora *et al.* [2017] this model was improved, among others, by using a mixture distribution of the form:

$$w_i = \alpha P(w_i) + (1-\alpha)\frac{1}{Z_S}\exp(\mathbf{w_i} \cdot \mathbf{c_S}) \tag{6}$$

with Z_S a normalizing constant. In this case, the maximum likelihood estimate of $\mathbf{c_S}$ corresponds to a weighted average of the word vectors, where the impact of the most common words is reduced. The estimated discourse vector $\hat{\mathbf{c_S}}$ will thus mostly be determined by the content words from S, and should thus capture the meaning of the sentence more clearly.

Intuitively, the model in (6) assumes that a fraction α of words in a typical sentence are non-informative words, which

are drawn from the background distribution $P(w_i)$. We can make this view explicit, which will be helpful for introducing our relation vector model. In particular, with each word position i we associate an indicator variable Z_i, where $Z_i = 1$ means that the word at that position has been drawn from the background distribution and $Z_i = 0$ otherwise. Let us furthermore consider a random variable W_i which corresponds to the word that has been sampled at position i. We now consider the following generative process $i \in \{1, ..., k\}$:

$$P(Z_i{=}1) = \alpha \quad (7)$$

$$P(W_i{=}w \,|\, Z_i{=}1) = P(w) \quad (8)$$

$$P(W_i{=}w \,|\, Z_i{=}0) \propto \exp(\mathbf{w} \cdot \mathbf{c_S}) \quad (9)$$

Note that $P(W_i{=}w_i)$ is then given by (6).

3.2 Relation Vectors

Our generative process is similar in spirit to (7)–(9), but we will consider indicator variables Z_i which can take four possible values: z_{ab}, z_a, z_b and z_*. Intuitively, $Z_i{=}z_{ab}$ means that the word at position i is characteristic of the relationship between a and b; $Z_i{=}z_a$ means that the word is characteristic of a but not of b, and vice versa for $Z_i{=}z_b$; finally $Z_i{=}z_*$ means that the word is unrelated to a and b. Crucially, while the probability that $Z_i{=}1$ is fixed for each i in (7), we will instead infer a distribution over $\{z_{ab}, z_a, z_b, z_*\}$ for each Z_i, using expectation-maximization (EM).

Each of the possible values z_{ab}, z_a, z_b, z_* will be associated with a corresponding discourse vector. For efficiency, the discourse vectors $\mathbf{c_a}, \mathbf{c_b}, \mathbf{c}_*$ will correspond to pre-defined vectors. Specifically, $\mathbf{c}_*$ is obtained by taking an average of all the word vectors from the vocabulary, weighted by their frequency in the considered corpus. Similarly, for each word a from the vocabulary, $\mathbf{c_a}$ is estimated by simply averaging all the words that appear in the context of a throughout the corpus. The discourse vector $\mathbf{c_{ab}}$, on the other hand, is treated as a latent vector whose parameters will be inferred.

In the following, we let $S = \{w_1, ..., w_k\}$ be the bag-of-words representation consisting of all words that occur between a and b in sentences where a appears before b in the given corpus[2]. We initialise the discourse vector $\mathbf{c_{ab}}$ as:

$$\mathbf{c_{ab}} = \text{norm}\,\big(\sum_{i=1}^{k} \mathbf{w_i}\big)$$

Using EM, we alternatingly (i) compute the posterior probabilities $P(Z_i{=}z \,|\, W_i{=}w_i, \mathbf{c}_z)$ for $z \in \{z_{ab}, z_a, z_b, z_*\}$ and (ii) refine our estimate of the discourse vector $\mathbf{c_{ab}}$ by maximizing expected likelihood. Specifically, the posterior can be computed as follows:

$$\begin{aligned} P(Z_i{=}z \,|\, W_i{=}w_i; \mathbf{c}_z) &\propto P(W_i{=}w_i | Z_i{=}z; \mathbf{c}_z) \cdot P(Z_i{=}z) \\ &\propto \exp(\mathbf{w_i} \cdot \mathbf{c_z}) \cdot \lambda_z \end{aligned}$$

where $\lambda_{ab}, \lambda_a, \lambda_b, \lambda_* \geq 0$ are hyper-parameters of our model. To update the discourse vector $\mathbf{c_{ab}}$, first note that the expected log-likelihood is given by:

$$\begin{aligned} &\mathbb{E}_Z\left[\log \prod_{i=1}^{k} P(W_i{=}w_i)\right] = \mathbb{E}_Z\left[\sum_{i=1}^{k} \log P(W_i{=}w_i)\right] \\ &= A + \sum_{i=1}^{k} \mathbf{w_i} \cdot \mathbf{c_{ab}} \cdot P(Z_i = z_{ab}) \end{aligned}$$

for some constant A and $Z = (Z_1, ..., Z_k)$. It can be shown that the (normalized) discourse vector $\mathbf{c_{ab}}$ which maximizes the latter expression is given by:

$$\mathbf{c_{ab}} = \text{norm}\left(\sum_{i=1}^{k} \mathbf{w_i} \cdot P(Z_i{=}z_{ab})\right) \quad (10)$$

Propagation to unseen pairs. As mentioned in Section 2, one of the main advantages of the models of Washio and Kato [2018b] and Joshi *et al.* [2019] is their ability of inferring relation embeddings even for pairs of words that never co-occur in the same sentence. To achieve this with our method, we propose the following simple fallback strategy for pairs (a, b) for which we cannot compute a relation vector directly. We compute the 5 nearest neighbours $a_1, ..., a_5$ of a and the 5 nearest neighbours $b_1, ..., b_5$ of b, using standard word vectors. Let V be the pairs of words for which we can construct a relation vector directly. We then estimate the relation vector $\hat{\mathbf{c}}_{\mathbf{ab}}$ for (a, b) as:

$$\text{norm}\Big(\sum_{1 \leq i,j \leq 5, (a_i, b_j) \in V} \cos(a, a_i) \cdot \cos(b, b_j) \cdot \mathbf{c_{a_i b_j}}\Big) \quad (11)$$

4 Pre-trained Relation Embeddings

To learn the relation vectors we use the English Wikipedia dump of January 2018, as in Joshi *et al.* [2019]. Preprocessing of the corpus includes lowercasing and tokenization. Multiwords (e.g. *United States*) are also considered following the same approach as in Mikolov *et al.* [2013]. We used 300-dimensional FastText word embeddings [Bojanowski *et al.*, 2017] for initialization. The number of iterations of the EM algorithm (see Section 3.2) is set to three, which we empirically found sufficient for the model to converge.

Vocabulary selection. With our method we could potentially learn relation embeddings for all co-occurring word pairs. However, this would be slow to learn and require excessive amounts of storage, and would thus be impractical for downstream applications. Therefore we learn relation vectors for a more focused set of word pairs. As with word embedding models, we select our core word-level vocabulary based on plain frequency, in this case selecting the top 100K most frequent words in the corpus. Then, for selecting the most relevant word pairs, it is desirable to consider those which co-occur often, and whose joint occurrence probability is as high as possible, but at the same time without exceedingly rewarding rare events. Thus, we sort our pair-level vocabulary via, first, a minimum co-occurrence of 25, and second, by scoring each pair with a smoothed version of PMI, where we exponentiate the context word term

[2] In Jameel *et al.* [2018] and Espinosa-Anke and Schockaert [2018], additional vectors were learned based on the words appearing before a and those appearing after b. We only focus on the middle words, as this leads to a reduced overall dimensionality, which makes the method faster and easier to use in downstream tasks.

salamander-amphibian			tooth-chew			netherlandish-flemish		
relative	**pair2vec**	**vecdiff**	**relative**	**pair2vec**	**vecdiff**	**relative**	**pair2vec**	**vecdiff**
cagayan-province	salamander-lizard	toad-amphibian	claws-dig	jaw-chew	jaw-chew	penance-reconciliation	babylonian-palestinian	overview-flemish
mold-piece	salamander-toad	frog-amphibian	mandible-chew	tooth-jawbone	mandible-chew	courier-messenger	polynesian-tahitian	antwerp-flemish
filipinos-asian	shearwater-seabird	lizard-amphibian	tooth-grind	swallow-chew	molar-chew	persian-pahlavi	tyrolean-austrian	dutch-flemish
mindano-island	amphibian-salamander	snake-amphibian	tooth-bite	vine-bud	tooth-gum	del pueblo-people	brickwork-flemish	hittite-anatolian
quilt-layer	mammal-amphibian	fish-amphibian	tick-host	tooth-lip	tooth-like	handbag-purse	dacian-romanian	overview-walloon

Table 1: Nearest neighbours for selected relation vectors. The main takeaway is that our proposed model provides greater abstractions, bringing closer similar relations from different domains, unlike vector differences and (to a lesser extent) Pair2Vec.

in the denominator to 0.5, similar as in Levy *et al.* [2015a]. Finally, to ensure a balanced vocabulary, we select a maximum of 100 co-occurring words (i.e. those with the highest smoothed-PMI scores) for each word in the vocabulary. This process yielded a pair vocabulary of 1,138,305 relation vectors of 300 dimensions. These pre-trained relation embeddings, along with the code to generate them, are available at https://github.com/pedrada88/relative.

Execution Time. As far as the training time is concerned, learning relation vectors for all pairs in the vocabulary took around a day on a standard desktop computer on CPU. This is considerably faster than competing systems. For instance, as explained in Section 3.2, SeVeN [Espinosa-Anke and Schockaert, 2018] is considerably slower, among others due to the use of sentence-level vector representations. GRV [Jameel *et al.*, 2018] requires solving a number of least-squares regression problems for each word pair, in addition to computing the ternary PMI score statistics, which also takes more time. For Pair2Vec [Joshi *et al.*, 2019], the original repository indicates a duration of 7-10 days to learn the relation model from the Wikipedia corpus, on unspecified hardware. This is due to the use of Bi-LSTM sentence encoding, pre-training, and compositional representation functions coupled with various types of sampling. For NLRA_{VO} [Washio and Kato, 2018b] no explicit information is available, but given that their model uses a similar LSTM-based encoding as Pair2Vec and moreover uses dependency contexts, we assume the training time to be even higher than for GRV.[3]

Nearest Neighbours. To gain an understanding of the intrinsic properties of our model we provide an analysis based on the nearest neighbours of selected relation vectors. In particular, Table 1 shows, for illustrative purposes, different semantic clusters generated by RELATIVE, Pair2Vec, and FastText word vector differences (VECDIFF), the latter being the most common approach to represent relations in the literature [Weeds *et al.*, 2014; Vylomova *et al.*, 2016]. We can see that interesting relational abstractions seem to emerge in our model, e.g., a clear *hypernymic* relation in the *salamander-amphibian* pair. Note that the nearest neighbours in the RELATIVE space, for this example, are word pairs for which a similar relation holds, but do not belong to the animal domain, unlike with PAIR2VEC and VECDIFF. Similarly, the *tooth-chew* pair intuitively captures the "is used for" relation while the *netherlandish-flemish* pair captures "is similar to". To gain further understanding, in Table 2 we show the word vectors that are most similar to our learned relation vectors, and compare them to the nearest neighbours of the vectors used for initialization (based on plain word vector averaging). We can clearly observe that the closest words to our relation vectors tend to be more characteristic of the relationship between the two words. For instance, the neighbors for *slovaks-ukranians* characterize the geo-spatial relationship, whereas the initialization vectors mostly include frequent function words. This is clearly related to using discourse vectors, which penalizes words that are not strictly relevant to the relation, but simply rather frequent overall.

slovaks-ukrainians		invasive species-natural		harvey-kansas	
relative	**init**	**relative**	**init**	**relative**	**init**
western	,	detrimental	the	,	county
eastern	and	potentially destabilizing	of	in	,
southern	in	detrimentally affected	which	county	in
southwestern	.	overlogging	be	townshippers	countys
southernwestern	at	thereby exacerbating	and	town	17-county

Table 2: Nearest neighbours (words) to selected relation vectors in the initialized and final versions of our RELATIVE model.

5 Intrinsic Evaluation: Lexical Semantics

In this section we evaluate our model on standard lexical semantics tasks. To this end, we use our latent variable model to learn relation embeddings for the word pairs occurring in the considered benchmarks, using the same experimental setting described in Section 4. In the case of word pairs that never occur in the same sentence in our corpus, we rely on the propagation explained in Section 3.2. We report results for our final model RELATIVE and for the plain-averaging based relation vectors $\mathbf{c_{ab}}$ that we used to initialize our model (see Section 3.2), which we will refer to as $\text{RELATIVE}_{\text{init}}$.

Comparison methods. We compare with the main relation embedding methods available in recent literature: 300-dimension Pair2Vec[4] [Joshi *et al.*, 2019] and GRV_{SI}[5][Jameel *et al.*, 2018], 600-d NLRA_{VO}[6] [Washio and Kato, 2018b] and

[3]There are some intrinsic differences among the models which should also be taken into consideration. In our case, as for SeVeN and GRV, relation embeddings are the final output. In the case of Pair2Vec and NLRA_{VO}, a neural model is learned which can be used to extract relation embeddings for any word pair.

[4]We used the pre-trained model available in https://github.com/mandarjoshi90/pair2vec. These models were trained on the same Wikipedia corpus used in our experiments (see Section 4).

[5]We used a variant of the original method, which computes relation vectors as a weighted average of the word vectors, similar to our method but using as weights the SI^2 scores from Section 4.1 of the original paper. We found this to perform better overall, while being faster to compute and more directly comparable with our model.

[6]No code available. We contacted the authors and they provided us with the embeddings used in the SemEval task. These are not

900-d SeVeN[7] [Espinosa-Anke and Schockaert, 2018].

Word pair encoding. To represent a given word pair (a, b), we concatenate the relation embeddings $\mathbf{c_{ab}}$ and $\mathbf{c_{ba}}$ to two vectors derived from the standard word embedding of a and b, namely the vector difference $\mathbf{b} - \mathbf{a}$ and the component-wise multiplication $\mathbf{a} \odot \mathbf{b}$, where adding the latter has been shown to improve performance in lexical semantics tasks [Vu and Shwartz, 2018]. As word embeddings, we use the same 300-dimensional FastText embeddings that are used for the initialization of our model (see Section 4). The simple concatenation of $\mathbf{b} - \mathbf{a}$ and $\mathbf{a} \odot \mathbf{b}$ is also used as a baseline.

5.1 Lexical Entailment

We evaluate our model on graded lexical entailment using the HyperLex dataset [Vulić *et al.*, 2017]. Instead of defining relations as binary, in this dataset each word pair (a, b) is provided with a score measuring the degree of membership of a to the category b, which makes this task well-suited for evaluating relation vectors. HyperLex contains 2,616 pairs with two evaluation protocols. For both protocols, training and test partitions are available. The first protocol (random split) includes vocabulary overlap between train and test instances while the second one (lexical) does not. We then train a linear SVM regression model on the encoded vectors from the word pairs of the training split, and the performance is evaluated in the test split, in terms of the Pearson (r) and Spearman (ρ) correlation values with respect to the gold standard.

Results. Table 3 summarizes the results. To interpret these results, it is important to highlight the finding of Levy *et al.* [2015b], who showed that supervised methods based on word embeddings are prone to memorize the training set, learning what they referred to as *prototypical hypernyms*. These are words that frequently occur in the training set, which leads the supervised model to consider them as super-classes, irrespective of the other word in the pair. To reliably evaluate the performance of different models, it is thus important to use the lexical split. We show results for the random split as well, to analyze which methods are most prone to overfitting. In Table 3, this effect can clearly be seen for all the baselines. Interestingly, the performance of Pair2Vec drops to below the word embedding baseline for the lexical split, suggesting that this model is particularly prone to the aforementioned memorization phenomenon. For the SeVeN model, a similar drop in performance between the random and lexical split is witnessed, again showing that this method overfits, although the overall performance of that method is higher. In contrast, our model is much more robust, obtaining by far the best performance on the lexical split. This clearly shows that our model is better at abstracting away from the particular words a and b when learning relation vectors.

5.2 Relation Classification and Similarity

Given a pre-defined set of relations and a pair of words, the **relation classification** task consists in selecting the relation that best describes the relationship between the two words. As test sets we used DiffVec [Vylomova *et al.*, 2016] and BLESS [Baroni and Lenci, 2011]. The DiffVec dataset contains 12,458 word pairs from a total of fifteen types of relations, e.g., hypernymy, event or cause-purpose. BLESS includes noun-noun relations such as hypernymy, meronymy, and co-hyponymy, including 13,258 and 6,629 instances for training and testing, respectively.[8] The relation classification task is treated as a multi-class classification problem, where the relations are encoded using word and relation embeddings as explained at the beginning of this section. For our experiments we train a linear SVM classifier directly on the vector differences, using 10-fold cross-validation in the case of DiffVec, and using the train-test splits in the case of BLESS.

		Lexical		Random	
	Model	r	ρ	r	ρ
Relation Emb	RELATIVE	**53.1**	**54.3**	56.8	58.4
	$\text{RELATIVE}_{\text{init}}$	42.7	43.9	**58.9**	**60.6**
	Pair2Vec	33.9	33.4	53.0	54.3
	GRV_{SI}	47.2	48.3	52.4	55.4
	SeVeN*	*47.6*	*46.9*	*61.2*	*62.7*
Word Emb	FastText	42.6	43.9	52.2	54.3

Table 3: Pearson (r) and Spearman (ρ) correlation performance on the lexical and random partitions of HyperLex. Models marked with * are not directly comparable (higher dimensionality).

Different relations may also have different degrees of prototypicality. For instance, for the relation "X is higher in the genealogical tree than Y", the pair *(son,father)* should be considered more prototypical than the pair *(son,uncle)*, even though both pairs might be considered to be instances of the relation. This property is measured in the the SemEval 2012 dataset on measuring degrees of **relational similarity** [Jurgens *et al.*, 2012]. We follow the experimental methodology of Jameel *et al.* [2018], which is based on the platinum ratings dataset.[9] In this dataset, 79 files, each one corresponding to a different relation, are provided. Note that this relational similarity task is about measuring the typicality of a word pair w.r.t. a fixed relation class, rather than measuring the degree of category membership of the first word in the pair with respect to the class represented by the second word as in HyperLex [Vulić *et al.*, 2017]. For each relation, we first split the associated relation files into two thirds for training and one third for evaluation. Then, as in the case of HyperLex, we train a linear SVM regression model on the encoded vectors of the ranked word pairs from the training split, and evaluate its performance on the test split.

Results. The results for the relation classification and similarity tasks are displayed in Table 4. As can be observed, relation embeddings are clearly useful in the relation classi-

directly comparable with our model as they have a higher number of dimensions (i.e. 600) and are trained on a different corpus. Nevertheless, we include their result in the SemEval task for completeness.

[7]Code available at https://bitbucket.org/luisespinosa/seven.

[8]The full version of DiffVec includes some instances and relations from BLESS and therefore the datasets exhibit overlap in a number of relations.

[9]In particular, we made use of the "Phase2AnswerScaled" data from the platinum rankings dataset, downloaded from https://sites.google.com/site/semeval2012task2/.

		DiffVec		BLESS		SE-12
	Model	**Acc**	**F1**	**Acc**	**F1**	r
Rels	RELATIVE	**87.0**	64.9	**94.0**	**91.9**	33.1
	$\text{RELATIVE}_{\text{init}}$	86.9	64.0	93.7	91.6	33.0
	Pair2Vec	86.7	**65.8**	92.3	89.5	32.7
	GRV_{SI}	86.6	64.5	92.9	90.6	**33.2**
	SeVeN*	*86.7*	*64.9*	*93.2*	*91.1*	*34.4*
	NRVA_{VO}*	-	-	-	-	*32.7*
Word	FastText	84.3	61.3	92.9	90.6	32.7

Table 4: Results using relation embeddings and/or word embeddings as features in relation classification (DiffVec and BLESS) and relation prototypicality (SemEval-12).

fication tasks (DiffVec and BLESS), as in all cases they outperform the word embedding baseline. Our method obtains the best results overall, although the differences are rather small and Pair2Vec achieves the best performance on DiffVec in terms of F1. What is particularly remarkable is that a simple averaging model such as $\text{RELATIVE}_{\text{init}}$ can achieve near-optimal results. Interestingly, the weighted averaging approach that is taken by the GRV_{SI} model actually performs worse than this simple averaging strategy. In the case of relational similarity (SemEval-12), the results are mixed, with all models performing similarly to the word embedding baseline. One of the main reasons for this behaviour seems to be the limited amount of training data, which includes only a few dozen examples per relation type.

6 Extrinsic Evaluation: Text Categorization

One useful way in which relation vectors can be used in downstream applications is by enriching word vectors. To generate the enriched representation of a given word a, we simply concatenate its standard word vector $\mathbf{a}$ with the average of all relation vectors of the form $\mathbf{c_{ab}}$ among our pretrained set of relation vectors (see Section 4). These enriched representations (or the normal word vectors in the case of the FastText baseline) constitute the input to a standard word-based C-BLSTM classification model, which in the first layer consists of a CNN, followed by a bidirectional LSTM.[10] We evaluate the quality of these enriched representations on *text categorization*. Given a text and a pre-defined set of labels (categories), the text categorization task consists of assigning the text to its most appropriate label. We specifically used the following standard datasets[11]: (1) `20news` [Lang, 1995] (including 20 fine-grained labels); (2) `reuters` [Lewis *et al.*, 2004] (we use its 8-label variant); (3) `bbc`[12] [Greene and Cunningham, 2006] (5 labels); and (4) `ohsumed` (23 labels).

The experimental results, summarized in Table 5, suggest that incorporating relational information is indeed beneficial for text categorization generally across the board. In particular, our proposed model RELATIVE is the only relation embedding technique that performs consistently better than the standard FastText embeddings for all datasets and measures. The simpler $\text{RELATIVE}_{\text{init}}$ also proves competitive, achieving the best overall results in the challenging Ohsumed dataset (specialized medical domain) in terms of macro-average F1.

	20News		Reuters		BBC		Ohsumed	
Model	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**
RELATIVE	**79.0**	**78.3**	**97.0**	**92.2**	**97.8**	96.1	42.6	36.1
$\text{RELATIVE}_{\text{init}}$	78.8	78.0	96.6	89.7	96.5	94.8	42.8	**36.8**
Pair2Vec	75.5	74.9	96.3	88.4	96.3	**96.2**	39.8	30.9
GRV_{SI}	78.7	78.0	96.7	88.8	97.0	95.3	**42.9**	36.2
SeVeN	78.5	77.9	96.9	90.8	96.8	95.0	42.8	35.7
FastText	78.5	77.7	96.8	89.0	96.4	94.7	41.8	34.2

Table 5: Accuracy and Macro-F1 results in text categorization.

7 Conclusions and Future Work

In this paper we have presented a new model for learning relation vectors from text corpora. The relation vectors learned from our model address two issues with respect to prior work. First, they are much faster to compute, even compared to averaging based models such as SeVeN. Second, by using a principled way to focus on the most relevant words only, we obtain relation vectors which are intuitively "purer", in the sense that they capture the relationship, without capturing the meaning of the individual words. This aspect can be observed in our nearest neighbours analysis and is further highlighted in the results on the lexical split of the lexical entailment task.

As future work, there are various ways in which our model can be further refined, such as the use of multi-prototype embeddings to obtain more relevant discourse vectors $\mathbf{c_a}$ and $\mathbf{c_b}$ in the case of ambiguous words, and the use of priors, e.g. based on (11), to alleviate sparsity issues. Finally, as the text categorization experiments show, there are straightforward ways in which relation vectors can be leveraged for standard downstream NLP tasks, which we have only begun to explore. We are planning to use our methods in other downstream tasks which can benefit from the background knowledge encoded in the relation embeddings, e.g. reading comprehension, following the line of Joshi *et al.* [2019].

Acknowledgments

Jose Camacho-Collados and Steven Schockaert were supported by ERC Starting Grant 637277. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

[Agichtein and Gravano, 2000] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proc. of Digital libraries*, pages 85–94, 2000.

[Arora *et al.*, 2016] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *TACL*, 4:385–399, 2016.

[10] Model developed in Keras: https://github.com/fchollet/keras.

[11] Preprocessed datasets (tokenized, lowercased) were downloaded at https://github.com/pedrada88/preproc-textclassification [Camacho-Collados and Pilehvar, 2018] or their official repository.

[12] For `bbc`, which does not include train-test splits, we performed 10-fold cross-validation.

[Arora *et al.*, 2017] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *Proc. of ICLR*, 2017.

[Banko *et al.*, 2007] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proc. of IJCAI*, pages 2670–2676, 2007.

[Baroni and Lenci, 2011] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proc. GEMS Workshop*, pages 1–10, 2011.

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017.

[Camacho-Collados and Pilehvar, 2018] Jose Camacho-Collados and Mohammad Taher Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. In *Proc. of Blackbox EMNLP Workshop*, 2018.

[Carlson *et al.*, 2010] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI*, pages 1306–1313, 2010.

[dos Santos *et al.*, 2015] Cícero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. In *Proc. of ACL*, pages 626–634, 2015.

[Espinosa-Anke and Schockaert, 2018] Luis Espinosa-Anke and Steven Schockaert. SeVeN: Augmenting word embeddings with unsupervised relation vectors. In *Proc. of COLING*, pages 2653–2665, 2018.

[Greene and Cunningham, 2006] Derek Greene and Pádraig Cunningham. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. of ICML*, pages 377–384, 2006.

[Hashimoto *et al.*, 2015] Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Task-oriented learning of word embeddings for semantic relation classification. In *Proc. of CoNLL*, 2015.

[Hearst, 1992] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. of COLING*, pages 539–545, 1992.

[Jameel *et al.*, 2018] Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. Unsupervised learning of distributional relation vectors. In *Proc. of ACL*, 2018.

[Joshi *et al.*, 2019] Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. pair2vec: Compositional word-pair embeddings for cross-sentence inference. In *Proc. of NAACL*, pages 3597–3608, 2019.

[Jurgens *et al.*, 2012] David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. of SemEval*, pages 356–364, 2012.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings*, pages 331–339, 1995.

[Levy *et al.*, 2015a] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225, 2015.

[Levy *et al.*, 2015b] Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. Do supervised distributional methods really learn lexical inference relations? In *Proc. of NAACL*, pages 970–976, 2015.

[Lewis *et al.*, 2004] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119, 2013.

[Riedel *et al.*, 2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proc. of HLT-NAACL*, pages 74–84, 2013.

[Shinyama and Sekine, 2006] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proc. of NAACL-HLT*, pages 304–311, 2006.

[Turney, 2005] Peter D. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI*, pages 1136–1141, 2005.

[Vu and Shwartz, 2018] Tu Vu and Vered Shwartz. Integrating multiplicative features into supervised distributional methods for lexical entailment. In *Proc. of *SEM*, 2018.

[Vulić *et al.*, 2017] Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835, 2017.

[Vylomova *et al.*, 2016] Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proc. of ACL*, pages 1671–1682, 2016.

[Washio and Kato, 2018a] Koki Washio and Tsuneaki Kato. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Proc. of NAACL*, pages 1123–1133, 2018.

[Washio and Kato, 2018b] Koki Washio and Tsuneaki Kato. Neural latent relational analysis to capture lexical semantic relations in a vector space. In *Proc. of EMNLP*, 2018.

[Weeds *et al.*, 2014] Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *Proc. of COLING*, pages 2249–2259, 2014.

[Xu *et al.*, 2015] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. of EMNLP*, pages 1785–1794, 2015.