Relational Word Embeddings

Jose Camacho-Collados **Luis Espinosa-Anke** **Steven Schockaert**
School of Computer Science and Informatics
Cardiff University, United Kingdom
{camachocolladosj,espinosa-ankel,schockaerts1}@cardiff.ac.uk

Abstract

While word embeddings have been shown to implicitly encode various forms of attributional knowledge, the extent to which they capture relational information is far more limited. In previous work, this limitation has been addressed by incorporating relational knowledge from external knowledge bases when learning the word embedding. Such strategies may not be optimal, however, as they are limited by the coverage of available resources and conflate similarity with other forms of relatedness. As an alternative, in this paper we propose to encode relational knowledge in a separate word embedding, which is aimed to be complementary to a given standard word embedding. This relational word embedding is still learned from co-occurrence statistics, and can thus be used even when no external knowledge base is available. Our analysis shows that relational word vectors do indeed capture information that is complementary to what is encoded in standard word embeddings.

1 Introduction

Word embeddings are paramount to the success of current natural language processing (NLP) methods. Apart from the fact that they provide a convenient mechanism for encoding textual information in neural network models, their importance mainly stems from the remarkable amount of linguistic and semantic information that they capture. For instance, the vector representation of the word *Paris* implicitly encodes that this word is a noun, and more specifically a capital city, and that it describes a location in France. This information arises because word embeddings are learned from co-occurrence counts, and properties such as being a capital city are reflected in such statistics. However, the extent to which relational knowledge (e.g. *Trump* was the successor of *Obama*) can be learned in this way is limited.

Previous work has addressed this by incorporating external knowledge graphs (Xu et al., 2014; Celikyilmaz et al., 2015) or relations extracted from text (Chen et al., 2016). However, the success of such approaches depends on the amount of available relational knowledge. Moreover, they only consider well-defined discrete relation types (e.g. *is the capital of*, or *is a part of*), whereas the appeal of vector space representations largely comes from their ability to capture subtle aspects of meaning that go beyond what can be expressed symbolically. For instance, the relationship between *popcorn* and *cinema* is intuitively clear, but it is more subtle than the assertion that "popcorn *is located at* cinema", which is how ConceptNet (Speer et al., 2017), for example, encodes this relationship[1].

In fact, regardless of how a word embedding is learned, if its primary aim is to capture similarity, there are inherent limitations on the kinds of relations they can capture. For instance, such word embeddings can only encode similarity preserving relations (i.e. similar entities have to be related to similar entities) and it is often difficult to encode that w is in a particular relationship while preventing the inference that words with similar vectors to w are also in this relationship; e.g. Bouraoui et al. (2018) found that both (*Berlin*,*Germany*) and (*Moscow*,*Germany*) were predicted to be instances of the capital-of relation due to the similarity of the word vectors for *Berlin* and *Moscow*. Furthermore, while the ability to capture word analogies (e.g. *king-man+woman≈queen*) emerged as a successful illustration of how word embeddings can encode some types of relational information (Mikolov et al., 2013b), the generalization of this interesting property has proven to be less successful than initially anticipated (Levy et al., 2014;

---

[1] http://conceptnet.io/c/en/popcorn

Linzen, 2016; Rogers et al., 2017).

This suggests that relational information has to be encoded separately from standard similarity-centric word embeddings. One appealing strategy is to represent relational information by learning, for each pair of related words, a vector that encodes how the words are related. This strategy was first adopted by Turney (2005), and has recently been revisited by a number of authors (Washio and Kato, 2018a; Jameel et al., 2018; Espinosa Anke and Schockaert, 2018; Washio and Kato, 2018b; Joshi et al., 2019). However, in many applications, word vectors are easier to deal with than vector representations of word pairs.

The research question we consider in this paper is whether it is possible to learn word vectors that capture relational information. Our aim is for such *relational word vectors* to be complementary to standard word vectors. To make relational information available to NLP models, it then suffices to use a standard architecture and replace normal word vectors by concatenations of standard and relational word vectors. In particular, we show that such relational word vectors can be learned directly from a given set of relation vectors.

2   Related Work

Relation Vectors. A number of approaches have been proposed that are aimed at learning relation vectors for a given set of word pairs (a,b), based on sentences in which these word pairs co-occur. For instance, Turney (2005) introduced a method called Latent Relational Analysis (LRA), which relies on first identifying a set of sufficiently frequent lexical patterns and then constructs a matrix which encodes for each considered word pair (a,b) how frequently each pattern P appears in between a and b in sentences that contain both words. Relation vectors are then obtained using singular value decomposition. More recently, Jameel et al. (2018) proposed an approach inspired by the GloVe word embedding model (Pennington et al., 2014) to learn relation vectors based on co-occurrence statistics between the target word pair (a, b) and other words. Along similar lines, Espinosa Anke and Schockaert (2018) learn relation vectors based on the distribution of words occurring in sentences that contain a and b by averaging the word vectors of these co-occurring words. Then, a conditional autoencoder is used to obtain lower-dimensional relation vectors.

Taking a slightly different approach, Washio and Kato (2018a) train a neural network to predict dependency paths from a given word pair. Their approach uses standard word vectors as input, hence relational information is encoded implicitly in the weights of the neural network, rather than as relation vectors (although the output of this neural network, for a given word pair, can still be seen as a relation vector). An advantage of this approach, compared to methods that explicitly construct relation vectors, is that evidence obtained for one word is essentially shared with similar words (i.e. words whose standard word vector is similar). Among others, this means that their approach can in principle model relational knowledge for word pairs that never co-occur in the same sentence. A related approach, presented in (Washio and Kato, 2018b), uses lexical patterns, as in the LRA method, and trains a neural network to predict vector encodings of these patterns from two given word vectors. In this case, the word vectors are updated together with the neural network and an LSTM to encode the patterns. Finally, similar approach is taken by the Pair2Vec method proposed in (Joshi et al., 2019), where the focus is on learning relation vectors that can be used for cross-sentence attention mechanisms in tasks such as question answering and textual entailment.

Despite the fact that such methods learn word vectors from which relation vectors can be predicted, it is unclear to what extent these word vectors themselves capture relational knowledge. In particular, the aforementioned methods have thus far only been evaluated in settings that rely on the predicted relation vectors. Since these predictions are made by relatively sophisticated neural network architectures, it is possible that most of the relational knowledge is still captured in the weights of these networks, rather than in the word vectors. Another problem with these existing approaches is that they are computationally very expensive to train; e.g. the Pair2Vec model is reported to need 7-10 days of training on unspecified hardware[2]. In contrast, the approach we propose in this paper is computationally much simpler, while resulting in relational word vectors that encode relational information more accurately than those of the Pair2Vec model in lexical semantics tasks, as we will see in Section 5.

Knowledge-Enhanced Word Embeddings. Sev-

---

[2] `github.com/mandarjoshi90/pair2vec`

eral authors have tried to improve word embeddings by incorporating external knowledge bases. For example, some authors have proposed models which combine the loss function of a word embedding model, to ensure that word vectors are predictive of their context words, with the loss function of a knowledge graph embedding model, to encourage the word vectors to additionally be predictive of a given set of relational facts (Xu et al., 2014; Celikyilmaz et al., 2015; Chen et al., 2016). Other authors have used knowledge bases in a more restricted way, by taking the fact that two words are linked to each other in a given knowledge graph as evidence that their word vectors should be similar (Faruqui et al., 2015; Speer et al., 2017). Finally, there has also been work that uses lexicons to learn word embeddings which are specialized towards certain types of lexical knowledge, such as hypernymy (Nguyen et al., 2017; Vulic and Mrksic, 2018), antonymy (Liu et al., 2015; Ono et al., 2015) or a combination of various linguistic constraints (Mrkšić et al., 2017).

Our method differs in two important ways from these existing approaches. First, rather than relying on an external knowledge base, or other forms of supervision, as in e.g. (Chen et al., 2016), our method is completely unsupervised, as our only input consists of a text corpus. Second, whereas existing work has focused on methods for improving word embeddings, our aim is to learn vector representations that are *complementary* to standard word embeddings.

3 Model Description

We aim to learn representations that are complementary to standard word vectors and are specialized towards relational knowledge. To differentiate them from standard word vectors, they will be referred to as *relational word vectors*. We write $\mathbf{e_w}$ for the relational word vector representation of w. The main idea of our method is to first learn, for each pair of closely related words w and v, a relation vector $\mathbf{r_{wv}}$ that captures how these words are related, which we discuss in Section 3.1. In Section 3.2 we then explain how we learn relational word vectors from these relation vectors.

3.1 Unsupervised Relation Vector Learning

Our goal here is to learn relation vectors for closely related words. For both the selection of the vocabulary and the method to learn relation vec-

tors we mainly follow the initialization method of Camacho-Collados et al. (2019, RELATIVE$_{\text{init}}$) except for an important difference explained below regarding the symmetry of the relations. Other relation embedding methods could be used as well, e.g., (Jameel et al., 2018; Washio and Kato, 2018b; Espinosa Anke and Schockaert, 2018; Joshi et al., 2019), but this method has the advantage of being highly efficient. In the following we describe this procedure for learning relation vectors: we first explain how a set of potentially related word pairs is selected, and then focus on how relation vectors $\mathbf{r_{wv}}$ for these word pairs can be learned.

Selecting Related Word Pairs. Starting from a vocabulary $\mathcal{V}$ containing the words of interest (e.g. all sufficiently frequent words), as a first step we need to choose a set $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ of potentially related words. For each of the word pairs in $\mathcal{R}$ we will then learn a relation vector, as explained below. To select this set $\mathcal{R}$, we only consider word pairs that co-occur in the same sentence in a given reference corpus. For all such word pairs, we then compute their strength of relatedness following Levy et al. (2015a) by using a smoothed version of pointwise mutual information (PMI), where we use 0.5 as exponent factor. In particular, for each word $w \in \mathcal{V}$, the set $\mathcal{R}$ contains all sufficiently frequently co-occurring pairs (w, v) for which v is within the top-100 most closely related words to w, according to the following score:

$$PMI_{0.5}(u, v) = \log\left(\frac{n_{wv} \cdot s_{**}}{n_{w*} \cdot s_{v*}}\right) \qquad (1)$$

where n_{wv} is the harmonically weighted[3] number of times the words w and v occur in the same sentence within a distance of at most 10 words, and:

$$n_{w*} = \sum_{u \in \mathcal{V}} n_{wu}; \quad s_{v*} = n_{v*}^{0.5}; \quad s_{**} = \sum_{u \in \mathcal{V}} s_{u*}$$

This smoothed variant of PMI has the advantage of being less biased towards infrequent (and thus typically less informative) words.

Learning Relation Vectors. In this paper, we will rely on word vector averaging for learning relation vectors, which has the advantage of being much faster than other existing approaches, and thus allows us to consider a higher number of word pairs (or a larger corpus) within a fixed

---

[3] A co-occurrence in which there are k words in between w and v then receives a weight of $\frac{1}{k+1}$.

time budget. Word vector averaging has moreover proven surprisingly effective for learning relation vectors (Weston et al., 2013; Hashimoto et al., 2015; Fan et al., 2015; Espinosa Anke and Schockaert, 2018), as well as in related tasks such as sentence embedding (Wieting et al., 2016).

Specifically, to construct the relation vector $\mathbf{r_{wv}}$ capturing the relationship between the words w and v we proceed as follows. First, we compute a bag of words representation $\{(w_1, f_1), ..., (w_n, f_n)\}$, where f_i is the number of times the word w_i occurs in between the words w and v in any given sentence in the corpus. The relation vector $\mathbf{r_{wv}}$ is then essentially computed as a weighted average:

$$\mathbf{r_{wv}} = norm\left(\sum_{i=1}^{n} f_i \cdot \mathbf{w_i}\right) \tag{2}$$

where we write $\mathbf{w_i}$ for the vector representation of w_i in some given pre-trained word embedding, and $norm(\mathbf{v}) = \frac{\mathbf{v}}{\|\mathbf{v}\|}$.

In contrast to other approaches, we do not differentiate between sentences where w occurs before v and sentences where v occurs before w. This means that our relation vectors are symmetric in the sense that $\mathbf{r_{wv}} = \mathbf{r_{vw}}$. This has the advantage of alleviating sparsity issues. While the directionality of many relations is important, the direction can often be recovered from other information we have about the words w and v. For instance, knowing that w and v are in a capital-of relationship, it is trivial to derive that "v is the capital of w", rather than the other way around, if we also know that w is a country.

3.2 Learning Relational Word Vectors

The relation vectors $\mathbf{r_{wv}}$ capture relational information about the word pairs in $\mathcal{R}$. The relational word vectors will be induced from these relation vectors by encoding the requirement that $\mathbf{e_w}$ and $\mathbf{e_v}$ should be predictive of $\mathbf{r_{wv}}$, for each $(w, v) \in \mathcal{R}$. To this end, we use a simple neural network with one hidden layer,[4] whose input is given by $(\mathbf{e_w} + \mathbf{e_v}) \oplus (\mathbf{e_w} \odot \mathbf{e_v})$, where we write $\oplus$ for vector concatenation and $\odot$ for the component-wise multiplication. Note that the input needs to be symmetric, given that our relation

---

[4]More complex architectures could be used, e.g., (Joshi et al., 2019), but in this case we decided to use a simple architecture as the main aim of this paper is to encode all relational information into the word vectors, not in the network itself.
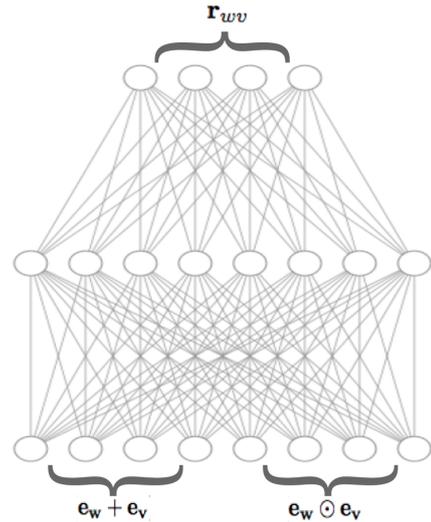

Figure 1: Relational word embedding architecture. At the bottom of the figure, the input layer is constructed from the relational word embeddings $\mathbf{e}_w$ and $\mathbf{e}_v$, which are the vectors to be learnt. As shown at the top, we aim to predict the target relation vector $\mathbf{r}_{wv}$.

vectors are symmetric, which makes the vector addition and component-wise multiplication two straightforward encoding choices. Figure 1 depicts an overview of the architecture of our model. The network is defined as follows:

$$\begin{aligned} \mathbf{i}_{wv} &= (\mathbf{e_w} + \mathbf{e_v}) \oplus (\mathbf{e_w} \odot \mathbf{e_v}) \\ \mathbf{h}_{wv} &= f(\mathbf{X}\mathbf{i}_{wv} + \mathbf{a}) \\ \mathbf{o}_{wv} &= f(\mathbf{Y}\mathbf{h}_{wv} + \mathbf{b}) \end{aligned} \tag{3}$$

for some activation function f. We train this network to predict the relation vector $\mathbf{r_{wv}}$, by minimizing the following loss:

$$\mathcal{L} = \sum_{(w,v)\in\mathcal{R}} \left(\mathbf{o}_{wv} - \mathbf{r}_{wv}\right)^2 \tag{4}$$

The relational word vectors $\mathbf{e_w}$ can be initialized using standard word embeddings trained on the same corpus.

4 Experimental Setting

In what follows, we detail the resources and training details that we used to obtain the relational word vectors.

Corpus and Word Embeddings. We followed the setting of Joshi et al. (2019) and used the English Wikipedia[5] as input corpus. Multiwords (e.g. *Manchester United*) were grouped together as a

---

[5]Tokenized and lowercased dump of January 2018.

single token by following the same approach described in Mikolov et al. (2013a). As word embeddings, we used 300-dimensional FastText vectors (Bojanowski et al., 2017) trained on Wikipedia with standard hyperparameters. These embeddings are used as input to construct the relation vectors $\mathbf{r}_{wv}$ (see Section 3.1),[6] which are in turn used to learn relational word embeddings $\mathbf{e}_w$ (see Section 3.2). The FastText vectors are additionally used as our baseline word embedding model.

Word pair vocabulary. As our core vocabulary $\mathcal{V}$, we selected the $100,000$ most frequent words from Wikipedia. To construct the set of word pairs $\mathcal{R}$, for each word from $\mathcal{V}$, we selected the 100 most closely related words (cf. Section 3.1), considering only consider word pairs that co-occur at least 25 times in the same sentence throughout the Wikipedia corpus. This process yielded relation vectors for 974,250 word pairs.

Training. To learn our relational word embeddings we use the model described in Section 3.2. The embedding layer is initialized with the standard FastText 300-dimensional vectors trained on Wikipedia. The method was implemented in PyTorch, employing standard hyperparameters, using ReLU as the non-linear activation function f (Equation 3). The hidden layer of the model was fixed to the same dimensionality as the embedding layer (i.e. 600). The stopping criterion was decided based on a small development set, by setting aside 1% of the relation vectors. Code to reproduce our experiments, as well as pre-trained models and details of the implementation such as other network hyperparameters, are available at https://github.com/pedrada88/rwe.

5 Experimental Results

A natural way to assess the quality of word vectors is to test them in lexical semantics tasks. However, it should be noted that relational word vectors behave differently from standard word vectors, and we should not expect the relational word vectors to be meaningful in unsupervised tasks such as semantic relatedness (Turney and Pantel, 2010). In particular, note that a high similarity between $\mathbf{e_w}$ and $\mathbf{e_v}$ should mean that relationships which hold for w have a high probability of holding for v as well. Words which are related, but not synonymous, may thus have very dissimilar relational word vectors. Therefore, we test our proposed models on a number of different supervised tasks for which accurately capturing relational information is crucial to improve performance.

Comparison systems. Standard FastText vectors, which were used to construct the relation vectors, are used as our main baseline. In addition, we also compare with the word embeddings that were learned by the Pair2Vec system[7] (see Section 2). We furthermore report the results of two methods which leverage knowledge bases to enrich FastText word embeddings: Retrofitting (Faruqui et al., 2015) and Attract-Repel (Mrkšić et al., 2017). Retrofitting exploits semantic relations from a knowledge base to re-arrange word vectors of related words such that they become closer to each other, whereas Attract-Repel makes use of different linguistic constraints to move word vectors closer together or further apart depending on the constraint. For Retrofitting we make use of WordNet (Fellbaum, 1998) as the input knowledge base, while for Attract-Repel we use the default configuration with all constraints from PPDB (Pavlick et al., 2015), WordNet and Babel-Net (Navigli and Ponzetto, 2012). All comparison systems are 300-dimensional and trained on the same Wikipedia corpus.

5.1 Relation Classification

Given a pre-defined set of relation types and a pair of words, the relation classification task consists in selecting the relation type that best describes the relationship between the two words. As test sets we used DiffVec (Vylomova et al., 2016) and BLESS[8] (Baroni and Lenci, 2011). The DiffVec dataset includes 12,458 word pairs, covering fifteen relation types including hypernymy, cause-purpose or verb-noun derivations. On the other hand, BLESS includes semantic relations such as hypernymy, meronymy, and co-hyponymy.[9] BLESS includes a train-test partition, with 13,258 and 6,629 word pairs, respectively. This task is treated as a multi-class classification problem

As a baseline model (Diff), we consider the usual representation of word pairs in terms of their vector differences (Fu et al., 2014; Roller et al.,

---

Encoding	Model	Reference	DiffVec				BLESS			
			Acc.	F1	Prec.	Rec.	Acc.	F1	Prec.	Rec.
Mult+Avg	RWE	(This paper)	**85.3**	**64.2**	65.1	**64.5**	**94.3**	**92.8**	**93.0**	**92.6**
	Pair2Vec	(Joshi et al., 2019)	85.0	64.0	65.0	**64.5**	91.2	89.3	88.9	89.7
	FastText	(Bojanowski et al., 2017)	84.2	61.4	62.6	61.9	92.8	90.4	90.7	90.2
	Retrofitting†	(Faruqui et al., 2015)	*86.1**	*64.6**	*66.6**	*64.5**	90.6	88.3	88.1	88.6
	Attract-Repel†	(Mrkšić et al., 2017)	*86.0**	*64.6**	*66.0**	*65.2**	91.2	89.0	88.8	89.3
Mult+Conc	Pair2Vec	(Joshi et al., 2019)	84.8	64.1	**65.7**	64.4	90.9	88.8	88.6	89.1
	FastText	(Bojanowski et al., 2017)	84.3	61.3	62.4	61.8	92.9	90.6	90.8	90.4
Diff (only)	FastText	(Bojanowski et al., 2017)	81.9	57.3	59.3	57.8	88.5	85.4	85.7	85.4

Table 1: Accuracy and macro-averaged F-Measure, precision and recall on BLESS and DiffVec. Models marked with † use external resources. The results with * indicate that WordNet was used for both the development of the model and the construction of the dataset. All models concatenate their encoded representations with the baseline vector difference of standard FastText word embeddings.

2014; Weeds et al., 2014), using FastText word embeddings. Since our goal is to show the complementarity of relational word embeddings with standard word vectors, for our method we concatenate the difference $\mathbf{w_j} - \mathbf{w_i}$ with the vectors $\mathbf{e_i} + \mathbf{e_j}$ and $\mathbf{e_i} \cdot \mathbf{e_j}$ (referred to as the Mult+Avg setting; our method is referred to as RWE). We use a similar representation for the other methods, simply replacing the relational word vectors by the corresponding vectors (but keeping the Fast-Text vector difference). We also consider a variant in which the FastText vector difference is concatenated with $\mathbf{w_i} + \mathbf{w_j}$ and $\mathbf{w_i} \cdot \mathbf{w_j}$, which offers a more direct comparison with the other methods. This goes in line with recent works that have shown how adding complementary features on top of the vector differences, e.g. multiplicative features (Vu and Shwartz, 2018), help improve the performance. Finally, for completeness, we also include variants where the average $\mathbf{e_i} + \mathbf{e_j}$ is replaced by the concatenation $\mathbf{e_i} \oplus \mathbf{e_j}$ (referred to as Mult+Conc), which is the encoding considered in Joshi et al. (2019).

For these experiments we train a linear SVM classifier directly on the word pair encoding, performing a 10-fold cross-validation in the case of DiffVec, and using the train-test splits of BLESS.

Results Table 1 shows the results of our relational word vectors, the standard FastText embeddings and other baselines on the two relation classification datasets (i.e. BLESS and DiffVec). Our model consistently outperforms the FastText embeddings baseline and comparison systems, with the only exception being the precision score for DiffVec. Despite being completely unsupervised, it is also surprising that our model manages to outperform the knowledge-enhanced embeddings

of Retrofitting and Attract-Repel in the BLESS dataset. For DiffVec, let us recall that both these approaches have the unfair advantage of having had WordNet as source knowledge base, used both to construct the test set and to enhance the word embeddings. In general, the improvement of RWE over standard word embeddings suggests that our vectors capture relations in a way that is compatible to standard word vectors (which will be further discussed in Section 6.2).

5.2 Lexical Feature Modelling

Standard word embedding models tend to capture semantic similarity rather well (Baroni et al., 2014; Levy et al., 2015a). However, even though other kinds of lexical properties may also be encoded (Gupta et al., 2015), they are not explicitly modeled. Based on the hypothesis that relational word embeddings should allow us to model such properties in a more consistent and transparent fashion, we select the well-known McRae Feature Norms benchmark (McRae et al., 2005) as testbed. This dataset[10] is composed of 541 words (or concepts), each of them associated with one or more features. For example, 'a *bear* is an *animal*', or 'a *bowl* is *round*'. As for the specifics of our evaluation, given that some features are only associated with a few words, we follow the setting of Rubinstein et al. (2015) and consider the eight features with the largest number of associated words. We carry out this evaluation by treating the task as a multi-class classification problem, where the labels are the word features. As in the previous task, we use a linear SVM classifier and perform 3-fold cross-validation. For each input word, the

---
[10]Downloaded from https://sites.google.com/site/kenmcraelab/norms-data

Model	McRae Feature Norms									QVEC
	Overall	metal	is_small	is_large	animal	is_edible	wood	is_round	is_long	
RWE	**55.2**	**73.6**	46.7	**45.9**	89.2	61.5	**38.5**	**39.0**	46.8	**55.4**
Pair2Vec	55.0	71.9	**49.2**	43.3	88.9	68.3	37.7	35.0	45.5	52.7
Retrofitting†	50.6	72.3	44.0	39.1	**90.6**	75.7	15.4	22.9	44.4	*56.8**
Attract-Repel†	50.4	73.2	44.4	33.3	88.9	71.8	31.1	24.2	35.9	*55.9**
FastText	54.6	72.7	48.4	45.2	87.5	63.2	33.3	**39.0**	47.8	54.6

Table 2: Results on the McRae feature norms dataset (Macro F-Score) and QVEC (correlation score). Models marked with † use external resources. The results with * indicate that WordNet was used for both the development of the model and the construction of the dataset.

word embedding of the corresponding feature is fed to the classifier concatenated with its baseline FastText embedding.

Given that the McRae Feature Norms benchmark is focused on nouns, we complement this experiment with a specific evaluation on verbs. To this end, we use the verb set of QVEC[11] (Tsvetkov et al., 2015), a dataset specifically aimed at measuring the degree to which word vectors capture semantic properties which has shown to strongly correlate with performance in downstream tasks such as text categorization and sentiment analysis. QVEC was proposed as an intrinsic evaluation benchmark for estimating the quality of word vectors, and in particular whether (and how much) they predict lexical properties, such as words belonging to one of the fifteen verb supersenses contained in WordNet (Miller, 1995). As is customary in the literature, we compute Pearson correlation with respect to these predefined semantic properties, and measure how well a given set of word vectors is able to predict them, with higher being better. For this task we compare the 300-dimensional word embeddings of all models (without concatenating them with standard word embeddings), as the evaluation measure only assures a fair comparison for word embedding models of the same dimensionality.

Results Table 2 shows the results on the McRae Feature Norms dataset[12] and QVEC. In the case of the McRae Feature Norms dataset, our relational word embeddings achieve the best overall results, although there is some variation for the individual features. These results suggest that attributional information is encoded well in our relational word embeddings. Interestingly, our results also suggest that Retrofitting and Attract-Repel,

which use pairs of related words during training, may be too naïve to capture the complex relationships proposed in these benchmarks. In fact, they perform considerably lower than the baseline Fast-Text model. On the other hand, Pair2Vec, which we recall is the most similar to our model, yields slightly better results than the FastText baseline, but still worse than our relational word embedding model. This is especially remarkable considering its much lower computational cost.

As far as the QVEC results are concerned, our method is only outperformed by Retrofitting and Attract-Repel. Nevertheless, the difference is minimal, which is surprising given that these methods leverage the same WordNet resource which is used for the evaluation.

6 Analysis

To complement the evaluation of our relational word vectors on lexical semantics tasks, in this section we provide a qualitative analysis of their intrinsic properties.

6.1 Word Embeddings: Nearest Neighbours

First, we provide an analysis based on the nearest neighbours of selected words in the vector space. Table 4 shows nearest neighbours of our relational word vectors and the standard FastText embeddings.[13] The table shows that our model captures some subtle properties, which are not normally encoded in knowledge bases. For example, geometric shapes are clustered together around the *sphere* vector, unlike in FastText, where more loosely related words such as "dimension" are found. This trend can easily be observed as well in the *philology* and *assassination* cases.

In the bottom row, we show cases where relational information is somewhat confused with col-

---

[11]https://github.com/ytsvetko/qvec
[12]Both *metal* and *wood* correspond to *made of* relations.

[13]Recall from Section 4 that both were trained on Wikipedia with the same dimensionality, i.e., 300.

SPHERE		PHILOLOGY		ASSASSINATION		DIVERSITY	
RWE	FastText	RWE	FastText	RWE	FastText	RWE	FastText
rectangle	spheres	metaphysics	philological	riot	assassinate	connectedness	cultural_diversity
conic	spherical	pedagogy	philologist	premeditate	attempt	openness	diverse
hexagon	dimension	docent	literature	bombing	attempts	creativity	genetic_diversity
INTERSECT		BEHAVIOUR		CAPABILITY		EXECUTE	
RWE	FastText	RWE	FastText	RWE	FastText	RWE	FastText
tracks	intersection	aggressive	behaviour	refueling	capabilities	murder	execution
northbound	bisect	detrimental	behavioural	miniaturize	capable	interrogation	executed
northwesterly	intersectional	distasteful	misbehaviour	positioning	survivability	incarcerate	summarily_executed

Table 3: Nearest neighbours for selected words in our relational word embeddings (RWE) and FastText embeddings

locationality, leading to undesired clusters, such as *intersect* being close in the space with "tracks", or *behaviour* with "aggressive" or "detrimental". These examples thus point towards a clear direction for future work, in terms of explicitly differentiating collocations from other relationships.

6.2 Word Relation Encoding

Unsupervised learning of analogies has proven to be one of the strongest selling points of word embedding research. Simple vector arithmetic, or pairwise similarities (Levy et al., 2014), can be used to capture a surprisingly high number of semantic and syntactic relations. We are thus interested in exploring *semantic clusters* as they emerge when encoding relations using our relational word vectors. Recall from Section 3.2 that relations are encoded using addition and pointwise multiplication of word vectors.

Table 4 shows, for a small number of selected word pairs, the top nearest neighbors that were unique to our 300-dimensional relational word vectors. Specifically, these pairs were not found among the top 50 nearest neighbors for the FastText word vectors of the same dimensionality, using the standard vector difference encoding. Similarly, we also show the top nearest neighbors that were unique to the FastText word vector difference encoding. As can be observed, our relational word embeddings can capture interesting relationships which go beyond what is purely captured by similarity. For instance, for the pair "innocent-naive" our model includes similar relations such as *vain-selfish*, *honest-hearted* or *cruel-selfish* as nearest neighbours, compared with the nearest neighbours of standard FastText embeddings which are harder to interpret.

Interestingly, even though not explicitly encoded in our model, the table shows some examples that highlight one property that arises often,

which is the ability of our model to capture cohyponyms as relations, e.g., *wrist-knee* and *anger-despair* as nearest neighbours of "shoulder-ankle" and "shock-grief", respectively. Finally, one last advantage that we highlight is the fact that our model seems to perform implicit disambiguation by balancing a word's meaning with its paired word. For example, the "oct-feb" relation vector correctly brings together other *month abbreviations* in our space, whereas in the FastText model, its closest neighbour is 'doppler-wheels', a relation which is clearly related to another sense of *oct*, namely its use as an acronym to refer to 'optical coherence tomography' (a type of x-ray procedure that uses the *doppler* effect principle).

6.3 Lexical Memorization

One of the main problems of word embedding models performing lexical inference (e.g. hypernymy) is lexical memorization. Levy et al. (2015b) found that the high performance of supervised distributional models in hypernymy detection tasks was due to a memorization in the training set of what they refer to as *prototypical hypernyms*. These prototypical hypernyms are general categories which are likely to be hypernyms (as occurring frequently in the training set) regardless of the hyponym. For instance, these models could equally predict the pairs *dog-animal* and *screen-animal* as hyponym-hypernym pairs. To measure the extent to which our model is prone to this problem we perform a controlled experiment on the lexical split of the HyperLex dataset (Vulić et al., 2017). This lexical split does not contain any word overlap between training and test, and therefore constitutes a reliable setting to measure the generalization capability of embedding models in a controlled setting (Shwartz et al., 2016). In HyperLex, each pair is provided by a score which measures the strength of the hypernymy relation.

INNOCENT-NAIVE		POLES-SWEDES		SHOULDER-ANKLE	
RWE	FastText	RWE	FastText	RWE	FastText
vain-selfish	murder-young	lithuanians-germans	polish-swedish	wrist-knee	oblique-ligament
honest-hearted	imprisonment-term	germans-lithuanians	poland-sweden	thigh-knee	pick-ankle_injury
cruel-selfish	conspiracy-minded	russians-lithuanians	czechoslovakia-sweden	neck-knee	suffer-ankle_injury
SHOCK-GRIEF		STRENGTHEN-TROPICAL_CYCLONE		OCT-FEB	
RWE	FastText	RWE	FastText	RWE	FastText
anger-despair	overcome-sorrow	intensify-tropical_cyclone	name-tropical_cyclones	aug-nov	doppler-wheels
anger-sorrow	overcome-despair	weaken-tropical_storm	bias-tropical_cyclones	sep-nov	scanner-read
anger-sadness	moment-sadness	intensify-tropical_storm	scheme-tropical_cyclones	nov-sep	ultrasound-baby

Table 4: Three nearest neighbours for selected word pairs using our relational word vector's relation encoding (RWE) and the standard vector difference encoding of FastText word embeddings. In each column only the word pairs which were on the top 50 NNs of the given model but not in the other are listed. Relations which include one word from the original pair were not considered.

Encoding	Model	r	ρ
Mult+Avg	RWE	**38.8**	**38.4**
	Pair2Vec	28.3	26.5
	FastText	37.2	35.8
	Retrofitting†	*29.5**	*28.9**
	Attract-Repel†	*29.7**	*28.9**
Mult+Conc	Pair2Vec	29.8	30.0
	FastText	35.7	33.3
Diff (only)	FastText	29.9	30.1

Table 5: Pearson (r) and Spearman (ρ) correlation on a subset of the HyperLex lexical split. Models marked with † use external resources. All models concatenate their encoded representations with the baseline vector difference of standard FastText word embeddings.

For these experiments we considered the same experimental setting as described in Section 4. In this case we only considered the portion of the HyperLex training and test sets covered in our vocabulary[14] and used an SVM regression model over the word-based encoded representations. Table 5 shows the results for this experiment. Even though the results are low overall (noting e.g. that results for the random split are in some cases above 50% as reported in the literature), our model can clearly generalize better than other models. Interestingly, methods such as Retrofitting and Attract-Repel perform worse than the FastText vectors. This can be attributed to the fact that these models have been mainly tuned towards similarity, which is a feature that loses relevance in this setting. Likewise, the relation-based embeddings of Pair2Vec do not help, probably due to the high-capacity of their model, which makes the word embeddings less informative.

---
[14]Recall from Section 4 that this vocabulary is shared by all comparison systems.

7 Conclusions

We have introduced the notion of relational word vectors, and presented an unsupervised method for learning such representations. Parting ways from previous approaches where relational information was either encoded in terms of relation vectors (which are highly expressive but can be more difficult to use in applications), represented by transforming standard word vectors (which capture relational information only in a limited way), or by taking advantage of external knowledge repositories, we proposed to learn an unsupervised word embedding model that is tailored specifically towards modelling relations. Our model is intended to capture knowledge which is complementary to that of standard similarity-centric embeddings, and can thus be used in combination.

We tested the complementarity of our relational word vectors with standard FastText word embeddings on several lexical semantic tasks, capturing different levels of relational knowledge. The evaluation indicates that our proposed method indeed results in representations that capture relational knowledge in a more nuanced way. For future work, we would be interested in further exploring the behavior of neural architectures for NLP tasks which intuitively would benefit from having access to relational information, e.g., text classification (Espinosa Anke and Schockaert, 2018; Camacho-Collados et al., 2019) and other language understanding tasks such as natural language inference or reading comprehension, in the line of Joshi et al. (2019).

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247.

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5.

Zied Bouraoui, Shoaib Jameel, and Steven Schockaert. 2018. Relation induction in word embeddings revisited. In *Proceedings of COLING*, pages 1627–1637.

Jose Camacho-Collados, Luis Espinosa-Anke, Shoaib Jameel, and Steven Schockaert. 2019. A latent variable model for learning distributional relation vectors. In *Proceedings of IJCAI*.

Asli Celikyilmaz, Dilek Hakkani-Tr, Panupong Pasupat, and Ruhi Sarikaya. 2015. Enriching word embeddings using knowledge graph for semantic tagging in conversational dialog systems. In *AAAI Spring Symposium*.

Jiaqiang Chen, Niket Tandon, Charles Darwis Hariman, and Gerard de Melo. 2016. Webbrain: Joint neural learning of large-scale commonsense knowledge. In *Proceedings of ISWC*, pages 102–118.

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

Miao Fan, Kai Cao, Yifan He, and Ralph Grishman. 2015. Jointly embedding relations and mentions for knowledge population. In *Proceedings of RANLP*, pages 186–191.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of NAACL*, pages 1606–1615.

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

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*, pages 1199–1209.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. Distributional vectors encode referential attributes. In *Proceedings of EMNLP*, pages 12–21.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of CoNLL*, pages 268–278.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018. Unsupervised learning of distributional relation vectors. In *Proceedings of ACL*, pages 23–33.

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2019. pair2vec: Compositional word-pair embeddings for cross-sentence inference. In *Proceedings of NAACL*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL*, pages 171–180.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of NAACL*.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 13–18.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of ACL*, pages 1501–1511.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

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

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using

monolingual and cross-lingual constraints. *Transactions of the Association of Computational Linguistics*, 5(1):309–324.

Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. 2017. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of EMNLP*, pages 233–243.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of NAACL-HLT*, pages 984–989.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Anna Rogers, Aleksandr Drozd, and Bofang Li. 2017. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (* SEM 2017)*, pages 135–148.

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

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of ACL*, pages 726–730.

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*, pages 2389–2398.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of AAAI*, pages 4444–4451.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of EMNLP*, pages 2049–2054.

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

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

Tu Vu and Vered Shwartz. 2018. Integrating multiplicative features into supervised distributional methods for lexical entailment. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 160–166.

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

Ivan Vulic and Nikola Mrksic. 2018. Specialising word vectors for lexical entailment. In *Proceedings of NAACL-HLT*, pages 1134–1145.

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

Koki Washio and Tsuneaki Kato. 2018a. Filling missing paths: Modeling co-occurrences of word pairs and dependency paths for recognizing lexical semantic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1123–1133.

Koki Washio and Tsuneaki Kato. 2018b. Neural latent relational analysis to capture lexical semantic relations in a vector space. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 594–600.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2249–2259.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of EMNLP*, pages 1366–1371.

John Wieting, Mohit Bansal, Kevin Gimple, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of ICLR*.

C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, and T.-Y. Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proc. CIKM*, pages 1219–1228.