Unsupervised Learning of Distributional Relation Vectors

Shoaib Jameel
School of Computing
Medway Campus
University of Kent, UK
M.S.Jameel@kent.ac.uk

Zied Bouraoui
Artois University and
CRIL CNRS
France
bouraoui@cril.fr

Steven Schockaert
School of Computer Science
and Informatics
Cardiff University, UK
schockaerts1@cardiff.ac.uk

Abstract

Word embedding models such as GloVe rely on co-occurrence statistics to learn vector representations of word meaning. While we may similarly expect that co-occurrence statistics can be used to capture rich information about the relationships between different words, existing approaches for modeling such relationships are based on manipulating pre-trained word vectors. In this paper, we introduce a novel method which directly learns relation vectors from co-occurrence statistics. To this end, we first introduce a variant of GloVe, in which there is an explicit connection between word vectors and PMI weighted co-occurrence vectors. We then show how relation vectors can be naturally embedded into the resulting vector space.

1 Introduction

Word embeddings are vector space representations of word meaning (Mikolov et al., 2013b; Pennington et al., 2014). A remarkable property of these models is that they capture various lexical relationships, beyond mere similarity. For example, (Mikolov et al., 2013b) found that analogy questions of the form "a is to b what c is to ?" can often be answered by finding the word d that maximizes $\cos(w_b - w_a + w_c, w_d)$, where we write w_x for the vector representation of a word x.

Intuitively, the word vector w_a represents a in terms of its most salient features. For example, w_{paris} implicitly encodes that Paris is located in France and that it is a capital city, which is intuitively why the 'capital of' relation can be modeled in terms of a vector difference. Other relationships, however, such as the fact that Macron succeeded Hollande as president of France, are unlikely to be captured by word embeddings. Relation extraction methods can discover such information by analyzing sentences that contain both of the words or entities involved (Mintz et al., 2009; Riedel et al., 2010; dos Santos et al., 2015), but they typically need a large number of training examples to be effective.

A third alternative, which we consider in this paper, is to characterize the relatedness between two words s and t by learning a relation vector r_{st} in an unsupervised way from corpus statistics. Among others, such vectors can be used to find word pairs that are similar to a given word pair (i.e. finding analogies), or to find the most prototypical examples among a given set of relation instances. They can also be used as an alternative to the aforementioned relation extraction methods, by subsequently training a classifier that uses the relation vectors as input, which might be particularly effective in cases where only limited amounts of training data are available (with the case of analogy finding from a single instance being an extreme example).

The most common unsupervised approach for learning relation vectors consists of averaging the embeddings of the words that occur in between s and t, in sentences that contain both (Weston et al., 2013; Fan et al., 2015; Hashimoto et al., 2015). While this strategy is often surprisingly effective (Hill et al., 2016), it is sub-optimal for two reasons. First, many of the words co-occurring with s and t will be semantically related to s or to t, but will not actually be descriptive for the relationship between s and t; e.g. the vector describing the relation between *Paris* and *France* should not be affected by words such as *eiffel* (which only relates to Paris). Second, it gives too much weight to stop-words, which cannot be addressed in a straightforward way as some stop-words are actually crucial for modeling relationships (e.g. prepositions such

as 'in' or 'of' or Hearst patterns (Indurkhya and Damerau, 2010)).

In this paper, we propose a method for learning relation vectors directly from co-occurrence statistics. We first introduce a variant of GloVe, in which word vectors can be directly interpreted as smoothed PMI-weighted bag-of-words representations. We then represent relationships between words as weighted bag-of-words representations, using generalizations of PMI to three arguments, and learn vectors that correspond to smoothed versions of these representations.

As far as the possible applications of our methodology is concerned, we imagine that relation vectors can be used in various ways to enrich the input to neural network models. As a simple example, in a question answering system, we could "annotate" mentions of entities with relation vectors encoding their relationship to the different words from the question. As another example, we could consider a recommendation system which takes advantage of vectors expressing the relationship between items that have been bought (or viewed) by a customer and other items from the catalogue. Finally, relation vectors should also be useful for knowledge completion, especially in cases where few training examples per relation type are given (meaning that neural network models could not be used) and where relations cannot be predicted from the already available knowledge (meaning that knowledge graph embedding methods could not be used, or are at least not sufficient).

2 Related Work

The problem of characterizing the relationship between two words has been studied in various settings. From a learning point of view, the most straightforward setting is where we are given labeled training sentences, with each label explicitly indicating what relationship is expressed in the sentence. This fully supervised setting has been the focus of several evaluation campaigns, including as part of ACE (Doddington et al., 2004) and at SemEval 2010 (Hendrickx et al., 2010). A key problem with this setting, however, is that labeled training data is hard to obtain. A popular alternative is to use known instances of the relations of interest as a form of distant supervision (Mintz et al., 2009; Riedel et al., 2010). Some authors have also considered unsupervised relation extraction methods (Shinyama and Sekine, 2006; Banko et al., 2007), in which case the aim is essentially to find clusters of patterns that express similar relationships, although these relationships may not correspond to the ones that are needed for the considered application. Finally, several systems have also used bootstrapping strategies (Brin, 1998; Agichtein and Gravano, 2000; Carlson et al., 2010), where a small set of instances are used to find extraction patterns, which are used to find more instances, which can in turn be used to find better extraction patterns, etc.

Traditionally, relation extraction systems have relied on a variety of linguistic features, such as lexical patterns, part-of-speech tags and dependency parses. More recently, several neural network architectures have been proposed for the relation extraction problem. These architectures rely on word embeddings to represent the words in the input sentence, and manipulate these word vectors to construct a relation vector. Some approaches simply represent the sentence (or the phrase connecting the entities whose relationship we want to determine) as a sequence of words, and use e.g. convolutional networks to aggregate the vectors of the words in this sequence (Zeng et al., 2014; dos Santos et al., 2015). Another possibility, explored in (Socher et al., 2012), is to use parse trees to capture the structure of the sentence, and to use recursive neural networks (RNNs) to aggregate the word vectors in a way which respects this structure. A similar approach is taken in (Xu et al., 2015), where LSTMs are applied to the shortest path between the two target words in a dependency parse. A straightforward baseline method is to simply take the average of the word vectors (Mitchell and Lapata, 2010). While conceptually much simpler, variants of this approach have obtained state-of-the-art performance for relation classification (Hashimoto et al., 2015) and a variety of tasks that require sentences to be represented as a vector (Hill et al., 2016).

Given the effectiveness of word vector averaging, in (Kenter et al., 2016) a model was proposed that explicitly tries to learn word vectors that generalize well when being averaged. Similarly, the model proposed in (Hashimoto et al., 2015) aims to produce word vectors that perform well for the specific task of relation classification. The ParagraphVector method from (Le and Mikolov, 2014) is related to the aforementioned approaches, but it explicitly learns a vector representation for each

paragraph along with the word embeddings. However, this method is computationally expensive, and often fails to outperform simpler approaches (Hill et al., 2016).

To the best of our knowledge, existing methods for learning relation vectors are all based on manipulating pre-trained word vectors. In contrast, we will directly learn relation vectors from corpus statistics, which will have the important advantage that we can focus on words that describe the interaction between the two words s and t, i.e. words that commonly occur in sentences that contain both s and t, but are comparatively rare in sentences that only contain s or only contain t.

Finally, note that our work is fundamentally different from Knowledge Graph Embedding (KGE) (Wang et al., 2014b), (Wang et al., 2014a), (Bordes et al., 2011) in at least two ways: (i) KGE models start from a structured knowledge graph whereas we only take a text corpus as input, and (ii) KGE models represent relations as geometric objects in the "entity embedding" itself (e.g. as translations, linear maps, combinations of projections and translations, etc), whereas we represent words and relations in different vector spaces.

3 Word Vectors as PMI Encodings

Our approach to relation embedding is based on a variant of the GloVe word embedding model (Pennington et al., 2014). In this section, we first briefly recall the GloVe model itself, after which we discuss our proposed variant. A key advantage of this variant is that it allows us to directly interpret word vectors in terms of the Pointwise Mutual Information (PMI), which will be central to the way in which we learn relation vectors.

3.1 Background

The GloVe model (Pennington et al., 2014) learns a vector w_i for each word i in the vocabulary, based on a matrix of co-occurrence counts, encoding how often two words appear within a given window. Let us write x_{ij} for the number of times word j appears in the context of word i in some text corpus. More precisely, assume that there are m sentences in the corpus, and let $\mathcal{P}_i^l \subseteq \{1, ..., n_l\}$ be the set of positions from the l^{th} sentence where the word i can be found (with n_l the length of the sentence). Then x_{ij} is defined as follows:

$$\sum_{l=1}^{m} \sum_{p \in \mathcal{P}_i^l} \sum_{q \in \mathcal{P}_j^l} \mathit{weight}(p, q)$$

where $\mathit{weight}(p, q) = \frac{1}{|p-q|}$ if $0 < |p - q| \leq W$, and $\mathit{weight}(p, q) = 0$ otherwise, where the window size W is usually set to 5 or 10.

The GloVe model learns for each word i two vectors w_i and $\tilde{w}_i$ by optimizing the following objective:

$$\sum_i \sum_{j: x_{ij} \neq 0} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \log x_{ij})^2$$

where f is a weighting function, aimed at reducing the impact of rare terms, and b_i and $\tilde{b}_j$ are bias terms. The GloVe model is closely related to the notion of pointwise mutual information (PMI), which is defined for two words i and j as $\mathit{PMI}(i, j) = \log\left(\frac{P(i,j)}{P(i)P(j)}\right)$, where $P(i, j)$ is the probability of seeing the words i and j if we randomly pick a word position from the corpus and a second word position within distance W from the first position. The PMI between i and j is usually estimated as follows:

$$\mathit{PMI}_X(i, j) = \log\left(\frac{x_{ij} x_{**}}{x_{i*} x_{*j}}\right)$$

where $x_{i*} = \sum_j x_{ij}$, $x_{*j} = \sum_i x_{ij}$ and $x_{**} = \sum_i \sum_j x_{ij}$. In particular, it is straightforward to see that after the reparameterization given by $b_i \mapsto b_i + \log x_{i*} - \log x_{**}$ and $b_j \mapsto b_j + \log x_{*j}$, the GloVe model is equivalent to

$$\sum_i \sum_{\substack{j \\ x_{ij} \neq 0}} f(x_{ij})(w_i \cdot \tilde{w}_j + b_i + \tilde{b}_j - \mathit{PMI}_X(i, j))^2 \tag{1}$$

3.2 A Variant of GloVe

In this paper, we will use the following variant of the formulation in (1):

$$\sum_i \sum_{j \in J_i} \frac{1}{\sigma_j^2} (w_i \cdot \tilde{w}_j + \tilde{b}_j - \mathit{PMI}_S(i, j))^2 \tag{2}$$

Despite its similarity, this formulation differs from the GloVe model in a number of important ways. First, we use smoothed frequency counts instead of the observed frequency counts x_{ij}. In particular, the PMI between words i and j is given as:

$$\mathit{PMI}_S(i, j) = \log\left(\frac{P(i, j)}{P(i)P(j)}\right)$$

where the probabilities are estimated as follows:

$$P(i) = \frac{x_{i*} + \alpha}{x_{**} + n\alpha} \qquad P(j) = \frac{x_{*j} + \alpha}{x_{**} + n\alpha}$$
$$P(i,j) = \frac{x_{ij} + \alpha}{x_{**} + n^2\alpha}$$

where $\alpha \geq 0$ is a parameter controlling the amount of smoothing and n is the size of the vocabulary. This ensures that the estimation of $PMI(i,j)$ is well-defined even in cases where $x_{ij} = 0$, meaning that we no longer have to restrict the inner summation to those j for which $x_{ij} > 0$. For efficiency reasons, in practice, we only consider a small subset of all context words j for which $x_{ij} = 0$, which is similar in spirit to the use of negative sampling in Skip-gram (Mikolov et al., 2013b). In particular, the set J_i contains each j such that $x_{ij} > 0$ as well as M uniformly[1] sampled context words j for which $x_{ij} = 0$, where we choose $M = 2 \cdot |\{j : x_{ij} > 0\}|$.

Second, following (Jameel and Schockaert, 2016), the weighting function $f(x_{ij})$ has been replaced by $\frac{1}{\sigma_j^2}$, where σ_j^2 is the residual variance of the regression problem for context word j, estimated follows:

$$\sigma_j^2 = \frac{1}{|J_j^{-1}|} \sum_{i \in J_j^{-1}} (w_i \cdot \tilde{w}_j + \tilde{b_j} - PMI_S(i,j))^2$$

with $J_j^{-1} = \{i : j \in J_i\}$. Since we need the word vectors to estimate this residual variance, we re-estimate σ_j^2 after every five iterations of the SGD optimization. For the first 5 iterations, where no estimation for σ_j^2 is available, we use the GloVe weighting function.

The use of smoothed frequency counts and residual variance based weighting make the word embedding model more robust for rare words. For instance, if w only co-occurs with a handful of other terms, it is important to prioritize the most informative context words, which is exactly what the use of the residual variance achieves, i.e. σ_j^2 is small for informative terms and large for stop words; see (Jameel and Schockaert, 2016). This will be important for modeling relations, as the relation vectors will often have to be estimated from very sparse co-occurrence counts.

[1]While the negative sampling method used in Skip-gram favors more frequent words, initial experiments suggested that deviating from a uniform distribution almost had no impact in our setting.

Finally, the bias term b_i has been omitted from the model in (2). We have empirically found that omitting this bias term does not affect the performance of the model, while it allows us to have a more direct connection between the vector w_i and the corresponding PMI scores.

3.3 Word Vectors and PMI

Let us define PMI_W as follows:

$$PMI_W(i,j) = w_i \cdot \tilde{w}_j + \tilde{b_j}$$

Clearly, when the word vectors are trained according to (2), it holds that $PMI_W(i,j) \approx PMI_S(i,j)$. In other words, we can think of the word vector w_i as a low-dimensional encoding of the vector $(PMI_S(i,1), ..., PMI_S(i,n))$, with n the number of words in the vocabulary. This view allows us to assign a natural interpretation to some word vector operations. In particular, the vector difference $w_i - w_k$ is commonly used as a model for the relationship between words i and k. For a given context word j, we have

$$(w_i - w_k) \cdot \tilde{w}_j = PMI_W(i,j) - PMI_W(k,j)$$

The latter is an estimation of $\log\left(\frac{P(i,j)}{P(i)P(j)}\right) - \log\left(\frac{P(k,j)}{P(k)P(j)}\right) = \log\left(\frac{P(j|i)}{P(j|k)}\right)$. In other words, the vector translation $w_i - w_k$ encodes for each context word j the (log) ratio of the probability of seeing j in the context of i and in the context of k, which is in line with the original motivation underlying the GloVe model (Pennington et al., 2014). In the following section, we will propose a number of alternative vector representations for the relationship between two words, based on generalizations of PMI to three arguments.

4 Learning Global Relation Vectors

We now turn to the problem of learning a vector r_{ik} that encodes how the source word i and target word k are related. The main underlying idea is that r_{ik} will capture which context words j are most closely associated with the word pair (i,k). Whereas the GloVe model is based on statistics about (*main word*, *context word*) pairs, here we will need statistics on (*source word*, *context word*, *target word*) triples. First, we discuss how co-occurrence statistics among three words can be expressed using generalizations of PMI to three arguments. Then we explain how this can be used to learn relation vectors in natural way.

4.1 Co-occurrence Statistics for Triples

Let $\mathcal{P}_i^l \subseteq \{1, ..., n_l\}$ again be the set of positions from the l^{th} sentence corresponding to word i. We define:

$$y_{ijk} = \sum_{l=1}^{m} \sum_{p \in \mathcal{P}_i^l} \sum_{q \in \mathcal{P}_j^l} \sum_{r \in \mathcal{P}_k^l} \textit{weight}(p, q, r)$$

where $\textit{weight}(p, q, r) = \max(\frac{1}{q-p}, \frac{1}{r-q})$ if $p < q < r$ and $r - p \leq W$, and $\textit{weight}(p, q, r) = 0$ otherwise. In other words, y_{ijk} reflects the (weighted) number of times word j appears between words i and k in a sentence in which i and k occur sufficiently close to each other, in that order. Note that by taking word order into account in this way, we will be able to model asymmetric relationships.

To model how strongly a context word j is associated with the word pair (i, k), we will consider the following two well-known generalizations of PMI to three arguments (Van de Cruys, 2011):

$$\textit{SI}^1(i, j, k) = \log \left(\frac{P(i, j)P(i, k)P(j, k)}{P(i)P(j)P(k)P(i, j, k)} \right)$$
$$\textit{SI}^2(i, j, k) = \log \left(\frac{P(i, j, k)}{P(i)P(j)P(k)} \right)$$

where $P(i, j, k)$ is the probability of seeing the word triple (i, j, k) when randomly choosing a sentence and three (ordered) word positions in that sentence within a window size of W. In addition we will also consider two ways in which PMI can be used more directly:

$$\textit{SI}^3(i, j, k) = \log \left(\frac{P(i, j, k)}{P(i, k)P(j)} \right)$$
$$\textit{SI}^4(i, j, k) = \log \left(\frac{P(i, k|j)}{P(i|j)P(k|j)} \right)$$

Note that $\textit{SI}^3(i, j, k)$ corresponds to the PMI between (i, k) and j, whereas $\textit{SI}^4(i, j, k)$ is the PMI between i and k conditioned on the fact that j occurs. The measures $\textit{SI}^3$ and $\textit{SI}^4$ are closely related to $\textit{SI}^1$ and $\textit{SI}^2$ respectively[2]. In particular, the following identities are easy to show:

$$\textit{PMI}(i, j) + \textit{PMI}(j, k) - \textit{SI}^1(i, j, k) = \textit{SI}^3(i, j, k)$$
$$\textit{SI}^2(i, j, k) - \textit{PMI}(i, j) - \textit{PMI}(j, k) = \textit{SI}^4(i, j, k)$$

[2]Note that probabilities of the form $P(i, j)$ or $P(i)$ here refer to marginal probabilities over ordered triples. In contrast, the PMI scores from the word embedding model are based on probabilities over unordered word pairs, as is common for word embeddings.

Using smoothed versions of the counts y_{ijk}, we can use the following probability estimates for $\textit{SI}^1(i, j, k)$–$\textit{SI}^4(i, j, k)$:

$$P(i, j, k) = \frac{y_{ijk} + \alpha}{y_{***} + n^3\alpha} \qquad P(i, j) = \frac{y_{ij*} + \alpha}{y_{***} + n^2\alpha}$$
$$P(i, k) = \frac{y_{i*k} + \alpha}{y_{***} + n^2\alpha} \qquad P(j, k) = \frac{y_{*jk} + \alpha}{y_{***} + n^2\alpha}$$
$$P(i) = \frac{y_{i**} + \alpha}{y_{***} + n\alpha} \qquad P(j) = \frac{y_{*j*} + \alpha}{y_{***} + n\alpha}$$
$$P(k) = \frac{y_{**k} + \alpha}{y_{***} + n\alpha}$$

where $y_{ij*} = \sum_k y_{ijk}$, and similar for the other counts. For efficiency reasons, the counts of the form y_{ij*}, y_{i*k} and y_{*jk} are pre-computed for all word pairs, which can be done efficiently due to the sparsity of co-occurrence counts (i.e. these counts will be 0 for most pairs of words), similarly to how to the counts x_{ij} are computed in GloVe. From these counts, we can also efficiently pre-compute the counts y_{i**}, y_{*j*}, y_{**k} and y_{***}. On the other hand, the counts y_{ijk} cannot be pre-computed, since the total number of triples for which $y_{ijk} \neq 0$ is prohibitively high in a typical corpus. However, using an inverted index, we can efficiently retrieve the sentences that contain the words i and k, and since this number of sentences is typically small, we can efficiently obtain the counts y_{ijk} corresponding to a given pair (i, k) whenever they are needed.

4.2 Relation Vectors

Our aim is to learn a vector r_{ik} that models the relationship between i and k. Computing such a vector for each pair of words (which co-occur at least once) is not feasible, given the number of triples (i, j, k) that would need to be considered. Instead, we first learn a word embedding, by optimizing (2). Then, fixing the context vectors $\tilde{w}_j$ and bias terms b_j, we learn a vector representation for a given pair (i, k) of interest by solving the following objective:

$$\sum_{j \in J_{i,k}} (r_{ik} \cdot \tilde{w}_j + \tilde{b_j} - \textit{SI}(i, j, k))^2 \qquad (3)$$

where $\textit{SI}$ refers to one of $\textit{SI}_S^1, \textit{SI}_S^2, \textit{SI}_S^3, \textit{SI}_S^4$. Note that (3) is essentially the counterpart of (1), where we have replaced the role of the PMI measure by SI. In this way, we can exploit the representations of the context words from the word embedding model for learning relation vectors. Note that the

factor $\frac{1}{\sigma_j^2}$ has been omitted. This is because words j that are normally relatively uninformative (e.g. stop words), for which σ_j^2 would be high, can actually be very important for characterizing the relationship between i and k. For instance, the phrase "X such as Y" clearly suggests a hyponomy relationship between X and Y, but both 'such' and 'as' would be associated with a high residual variance σ_j^2. The set $J_{i,k}$ contains every j for which $y_{ijk} > 0$ as well as a random sample of m words for which $y_{ijk} = 0$, where $m = 2 \cdot |\{j : y_{ijk} > 0|$. Note that because $\tilde{w}_j$ is now fixed, (3) is a linear least squares regression problem, which can be solved exactly and efficiently.

The vector r_{ik} is based on words that appear between i and k. In the same way, we can learn a vector s_{ik} based on the words that appear before i and a vector t_{ik} based on the words that appear after k, in sentences where i occurs before k. Furthermore, we also learn vectors r_{ki}, s_{ki} and t_{ki} from the sentences where k occurs before i. As the final representation R_{ik} of the relationship between i and k, we concatenate the vectors $r_{ik}, r_{ki}, s_{ik}, s_{ki}, t_{ik}, t_{ki}$ as well as the word vectors w_i and w_k. We write R_{ik}^l to denote the vector that results from using measure SI^l ($l \in \{1, 2, 3, 4\}$).

5 Experimental Results

In our experiments, we have used the Wikipedia dump from November 2nd, 2015, which consists of 1,335,766,618 tokens. We have removed punctuations and HTML/XML tags, and we have lowercased all tokens. Words with fewer than 10 occurrences have been removed from the corpus. To detect sentence boundaries, we have used the Apache sentence segmentation tool. In all our experiments, we have set the number of dimensions to 300, which was found to be a good choice in previous work, e.g. (Pennington et al., 2014). We use a context window size W of 10 words. The number of iterations for SGD was set to 50. For our model, we have tuned the smoothing parameter α based on held-out tuning data, considering values from $\{0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001\}$. We have noticed that in most of the cases the value of α was automatically selected as 0.00001. To efficiently compute the triples, we have used the Zettair[3] retrieval engine.

As our main baselines, we use three popular unsupervised methods for constructing relation vectors. First, *Diff* uses the vector difference $w_k - w_i$, following the common strategy of modeling relations as vector differences, as e.g. in (Vylomova et al., 2016). Second, *Conc* uses the concatenation of w_i and w_k. This model is more general than *Diff* but it uses twice as many dimensions, which may make it harder to learn a good classifier from few examples. The use of concatenations is popular e.g. in the context of hypernym detection (Baroni et al., 2012). Finally, *Avg* averages the vector representations of the words occurring in sentences that *Diff*, contain i and k. In particular, let r_{ik}^{avg} be obtained by averaging the word vectors of the context words appearing between i and k for each sentence containing i and k (in that order), and then averaging the vectors obtained from each of these sentences. Let s_{ik}^{avg} and t_{ik}^{avg} be similarly obtained from the words occurring before i and the words occurring after k respectively. The considered relation vector is then defined as the concatenation of r_{ik}^{avg}, r_{ki}^{avg}, s_{ik}^{avg}, s_{ki}^{avg}, t_{ik}^{avg}, t_{ki}^{avg}, w_i and w_k. The *Avg* will allow us to directly compare how much we can improve relation vectors by deviating from the common strategy of averaging word vectors.

5.1 Relation Induction

In the relation induction task, we are given word pairs $(s_1, t_1), ..., (s_k, t_k)$ that are related in some way, and the task is to decide for a number of test examples (s, t) whether they also have this relationship. Among others, this task was considered in (Vylomova et al., 2016), and a ranking version of this task was studied in (Drozd et al., 2016). As test sets we use the Google Analogy Test Set (Mikolov et al., 2013a), which contains instances of 14 different types of relations, and the DiffVec dataset, which was introduced in (Vylomova et al., 2016). This dataset contains instances of 36 different types of relations[4]. Note that both datasets contain a mix of semantic and syntactic relations.

In our evaluation, we have used 10-fold cross-validation (or leave-one-out for relations with fewer than 10 instances). In the experiments, we consider for each relation in the test set a separate binary classification task, which was found to be considerably more challenging than a multi-class classification setting in (Vylomova et al., 2016). To generate negative examples in the training data (resp. test data), we have used three strategies, fol-

[3] http://www.seg.rmit.edu.au/zettair/

[4] Note that in contrast to (Vylomova et al., 2016) we use all 36 relations from this dataset, including those with very few instances.

Table 1: Results for the relation induction task.

Google Analogy							
	Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
Acc	90.0	89.0	89.9	90.0	**92.3**	90.9	90.4
Pre	81.6	78.7	80.8	79.9	87.1	83.2	81.1
Rec	82.6	83.9	83.9	86.0	84.8	84.8	85.5
F1	82.1	81.2	82.3	82.8	**85.9**	84.0	83.3
DiffVec							
	Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
Acc	29.5	28.9	29.7	29.7	**31.3**	30.4	30.1
Pre	19.6	18.7	20.4	21.5	22.9	21.9	22.3
Rec	23.8	22.9	23.7	24.5	25.7	25.3	22.9
F1	21.5	20.6	21.9	22.4	**24.2**	23.5	22.6

lowing (Vylomova et al., 2016). First, for a given positive example (s, t) of the considered relation, we add (t, s) as a negative example. Second, for each positive example (s, t), we generate two negative examples (s, t_1) and (s, t_2) by randomly selecting two tail words t_1, t_2 from the other training (resp. test) examples of the same relation. Finally, for each positive example, we also generate a negative example by randomly selecting two words from the vocabulary. For each relation, we then train a linear SVM classifier. To set the parameters of the SVM, we initially use 25% of the training data for tuning, and then retrain the SVM with the optimal parameters on the full training data.

The results are summarized in Table 1 in terms of accuracy and (macro-averaged) precision, recall and F1 score. As can be observed, our model outperforms the baselines on both datasets, with the R^2_{ik} variant outperforming the others. To analyze the benefit of our proposed word embedding variant, Table 2 shows the results that were obtained when we use a standard word embedding model. In particular, we show results for the standard GloVe model, SkipGram and the Continuous Bag of Words (CBOW) model. As can be observed, our variant leads to better results than the original GloVe model, even for the baselines. The difference is particularly noticeable for DiffVec. The difference is also larger for our relation vectors than for the baselines, which is expected as our method is based on the assumption that context word vectors can be interpreted in terms of PMI scores, which is only true for our variant.

Similar as in the GloVe model, the context words in our model are weighted based on their distance to the nearest target word. Table 3 shows the results of our model without this weighting, for the relation induction task. Comparing these results with those in Table 1 shows that the weighting scheme indeed leads to a small improvement (except for the accuracy of R^1_{ik} for DiffVec). Similarly, in Table 3, we show what happens if the relation vectors s_{ik}, s_{ki}, t_{ik} and t_{ki} are omitted. In other words, for the results in Table 3, we only use context words that appear between the two target words. Again, the results are worse than those in Table 1 (with the accuracy of R^1_{ik} for DiffVec again being an exception), although the differences are very small in this case. While including the vectors s_{ik}, s_{ki}, t_{ik}, t_{ki} should be helpful, it also significantly increases the dimensionality of the vectors R^l_{ik}. Given that the number of instances per relation is typically quite small for this task, this can also make it harder to learn a suitable classifier.

5.2 Measuring Degrees of Prototypicality

Instances of relations can often have different degrees of prototypicality. For example, for the relation "X characteristically makes the sound Y", the pair (*dog*,*bark*) should be considered more prototypical than the pair (*floor*,*squeak*), even though both pairs might be considered to be instances of the relation (Jurgens et al., 2012). A suitable relation vector should allow us to rank word pairs according to how prototypical they are as instances of that relation. We evaluate this ability using a dataset that was produced in the aftermath of SemEval 2012 Task 2. In particular, we have used the "Phase2AnswerScaled" data from the platinum rankings dataset, which is available from the SemEval 2012 Task 2 website[5]. In this dataset, 79 ranked list of word pairs are provided, each of which corresponds to a particular relation. For each relation, we first split the associated ranking into 60% training, 20% tuning, and 20% testing (i.e. we randomly select 60% of the word pairs and use their ranking as training data, and similar for tuning and test data). We then train a linear SVM regression model on the ranked word pairs. Note that this task slightly differs from the task that was considered at SemEval 2012, to allow us to use an SVM based model for consistency with the rest of the paper.

We report results using Spearman's ρ in Table 4. Our model again outperforms the baselines, with R^2_{ik} again being the best variant. Interestingly, in this case, the Avg baseline is considerably stronger than Diff and Conc. Intuitively, we

[5]https://sites.google.com/site/semeval2012task2/download

Table 2: Results for the relation induction task using alternative word embedding models.

	GloVe				SkipGram				CBOW			
	Google		DiffVec		Google		DiffVec		Google		DiffVec	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Diff	90.0	81.9	21.2	13.9	89.8	81.9	21.7	14.5	89.9	82.1	17.4	9.7
Conc	88.9	80.4	20.2	11.9	89.2	81.6	20.5	12.0	89.1	81.1	16.4	7.7
Avg	89.8	82.1	21.4	13.9	90.2	82.4	21.8	14.4	89.8	82.2	17.5	10.0
R^1_{ik}	89.7	81.7	20.9	12.5	89.4	81.2	21.1	12.3	89.8	81.9	17.2	9.2
R^2_{ik}	90.0	82.8	21.2	13.4	89.1	81.3	21.1	12.9	90.2	82.4	17.7	10.0
R^3_{ik}	90.0	82.3	20.0	11.2	89.5	81.1	20.5	12.3	89.5	81.1	17.2	9.6
R^4_{ik}	90.0	82.5	20.0	11.4	88.9	80.8	20.6	12.1	90.5	82.2	17.1	8.4

Table 3: Relation induction without position weighting (left) and without the relation vectors s_{ik} and t_{ik} (right).

	Google		DiffVec	
	Acc	F1	Acc	F1
R^1_{ik}	89.7	82.4	30.2	22.2
R^2_{ik}	91.0	83.4	30.8	24.1
R^3_{ik}	90.4	83.2	30.1	22.3
R^4_{ik}	90.2	82.9	29.1	21.2

	Google		DiffVec	
	Acc	F1	Acc	F1
R^1_{ik}	90.0	82.5	29.9	22.3
R^2_{ik}	92.3	85.8	31.2	24.2
R^3_{ik}	90.5	83.2	30.2	23.0
R^4_{ik}	90.3	83.1	29.8	22.3

Table 4: Results for measuring degrees of prototypicality (Spearman $\rho \times 100$).

Diff	Conc	Avg	R^1_{ik}	R^2_{ik}	R^3_{ik}	R^4_{ik}
17.3	16.7	21.1	22.7	**23.9**	21.8	22.2

might indeed expect that this ranking problem requires a more fine-grained representation than the relation induction setting. Note that the Diff representations were found to achieve near state-of-the-art performance on a closely related task in (Zhila et al., 2013). The only model that was found to perform (slightly) better was a hybrid model, combining Diff representations with linguistic patterns (inspired by (Rink and Harabagiu, 2012)) and lexical databases, among others.

5.3 Relation Extraction

Finally, we consider the problem of relation extraction from a text corpus. Specifically, we consider the task proposed in (Riedel et al., 2010), which is to extract (subject,predicate,object) triples from the New York Times (NYT) corpus. Rather than having labelled sentences as training data, the task requires using the existing triples from Freebase as a form of distant supervision, i.e. for some pairs of entities we know some of the

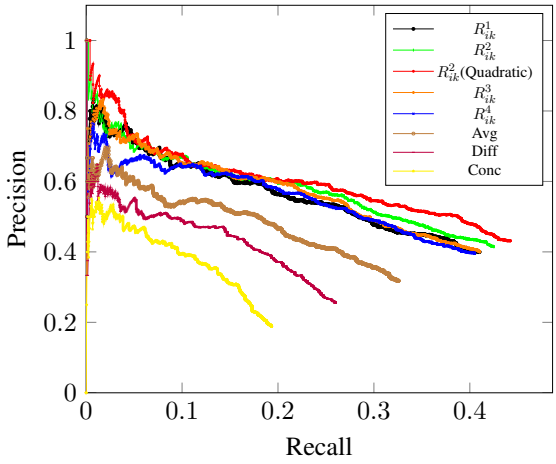

Figure 1: Results for the relation extraction from the NYT corpus: comparison with the main baselines.

relations that hold between them, but not which sentences assert these relationships (if any). To be consistent with published results for this task, we have used a word embedding that was trained from the NYT corpus[6], rather than Wikipedia (using the same preprocessing and set-up). We have used the training and test data that was shared publicly for this task[7], which consist of sentences from articles published in 2005-2006 and in 2007, respectively. Each of these sentences contains two entities, which are already linked to Freebase. We learn relation vectors from the sentences in the training and test sets, and learn a linear SVM classifier based on the Freebase triples that are available in the training set. Initially, we split the training data into 75% training and 25% tuning to find the optimal parameters of the linear SVM model. We tuned the parameters for each test fold separately. For each test fold, we used 25% of the 9 training folds as tuning data. After the optimal

[6]https://catalog.ldc.upenn.edu/LDC2008T19
[7]http://iesl.cs.umass.edu/riedel/ecml/

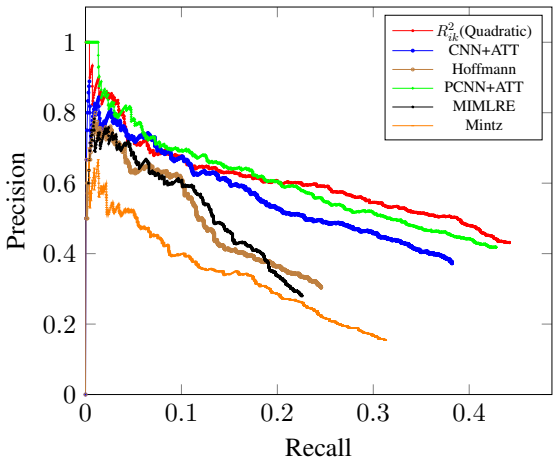

Figure 2: Results for the relation extraction from the NYT corpus: comparison with state-of-the-art neural network models.

parameters have been determined, we retrain the model on the full training data, and apply it on the test fold. We used this approach (rather than e.g. fixing a train/tune/test split) because the total number of examples for some of the relations is very small. After tuning, we re-train the SVM models on the full training data. As the number of training examples is larger for this task, we also consider SVMs with a quadratic kernel.

Following earlier work on this task, we report our results on the test set as a precision-recall graph in Figure 1. This shows that the best performance is again achieved by R^2_{ik}, especially for larger recall values. Furthermore, using a quadratic kernel (only shown for R^2_{ik}) outperforms the linear SVM models. Note that the differences between the baselines are more pronounced in this task, with Avg being clearly better than Diff, which is in turn better than Conc. For this relation extraction task, a large number of methods have already been proposed in the literature, with variants of convolutional neural network models with attention mechanisms achieving state-of-the-art performance[8]. A comparison with these models[9] is shown in Figure 2. The performance of R^2_{ik} is comparable with the state-of-the-art PCNN+ATT model (Lin et al., 2016), outperforming it for larger recall values. This is remarkable, as our model is conceptually much simpler, and has not been specifically tuned for this task. For instance, it could easily be improved by incorporating the attention mechanism from the PCNN+ATT model to focus the relation vectors on the considered task. Similarly, we could consider a supervised variant of (3), in which a learned relation-specific weight is added to each term.

[8]Note that such models would not be suitable for the evaluation tasks in Sections 5.1 and 5.2, due to the very limited number of training examples.

[9]Results for the neural network models have been obtained from https://github.com/thunlp/TensorFlow-NRE/tree/master/data.

6 Conclusions

We have proposed an unsupervised method which uses co-occurrences statistics to represent the relationship between a given pair of words as a vector. In contrast to neural network models for relation extraction, our model learns relation vectors in an unsupervised way, which means that it can be used for measuring relational similarities and related tasks. Moreover, even in (distantly) supervised tasks (where we need to learn a classifier on top of the unsupervised relation vectors), our model has proven competitive with state-of-the-art neural network models. Compared to approaches that rely on averaging word vectors, our method is able to learn more faithful representations by focusing on the words that are most strongly related to the considered relationship.

Acknowledgments

This work was supported by ERC Starting Grant 637277. Experiments in this work were performed using the computational facilities of the Advanced Research Computing at Cardiff (ARCCA) Division, Cardiff University and University of Kent High Performance Computing (HPC) services.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital libraries*. pages 85–94.

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

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

A. Bordes, J. Weston, R. Collobert, and Y. Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *International Workshop on The World Wide Web and Databases*. pages 172–183.

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

George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie Strassel, and Ralph M. Weischedel. 2004. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *Proc. LREC*.

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

Aleksandr Drozd, Anna Gladkova, and Satoshi Matsuoka. 2016. Word embeddings, analogies, and machine learning: Beyond king - man + woman = queen. In *Proc. COLING*. pages 3519–3530.

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

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

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proc. SemEval*. pages 33–38.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proc. NAACL-HLT*. pages 1367–1377.

Nitin Indurkhya and Fred J Damerau. 2010. *Handbook of natural language processing*, volume 2. CRC Press.

Shoaib Jameel and Steven Schockaert. 2016. D-GloVe: A feasible least squares model for estimating word embedding densities. In *Proc. COLING*. pages 1849–1860.

David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proc. *SEM*. pages 356–364.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: optimizing word embeddings for sentence representations. In *Proc. ACL*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. ICML*. pages 1188–1196.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proc. ACL*.

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

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc. ACL*. pages 1003–1011.

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

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

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proc. ECML/PKDD*. pages 148–163.

Bryan Rink and Sanda Harabagiu. 2012. UTD: Determining relational similarity using lexical patterns. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. pages 413–418.

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

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proc. EMNLP*. pages 1201–1211.

Tim Van de Cruys. 2011. Two multivariate generalizations of pointwise mutual information. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*. pages 16–20.

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 *Proc. ACL*.

Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014a. Knowledge graph and text jointly embedding. In *EMNLP*. pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*. pages 1112–1119.

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

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

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proc. COLING*. pages 2335–2344.

Alisa Zhila, Wen-tau Yih, Christopher Meek, Geoffrey Zweig, and Tomas Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1000–1009.