Abstract

Lexical relations describe how meanings of terms relate to each other. Typical relations include hypernymy, synonymy, meronymy, etc. Automatic distinction of lexical relations is vital for NLP applications, and is also challenging due to the lack of contextual signals to discriminate between such relations. In this work, we present a neural representation learning model to distinguish lexical relations among term pairs based on Hyperspherical Relation Embeddings (SphereRE). Rather than learning embeddings for individual terms, the model learns representations of relation triples by mapping them to the hyperspherical embedding space, where relation triples of different lexical relations are well separated. We further introduce a Monte-Carlo based sampling and learning algorithm to train the model via transductive learning. Experiments over several benchmarks confirm SphereRE outperforms state-of-the-arts.

1 Introduction

Lexical relations are relations between terms in lexicons. Types of lexical relations include hypernymy, synonymy, meronymy, etc. Such relations are treated as key resources for various NLP applications, e.g., question answering (Yang et al., 2017), taxonomy induction (Shen et al., 2018), machine translation (Zhang et al., 2018), natural language inference (Inkpen et al., 2018), lexical database construction (Speer et al., 2017), etc.

Due to its importance, automatic acquisition of lexical relations is a research focus in NLP. In early years, lexical relations in WordNet were manually compiled by linguists (Miller, 1995). Recently, path-based and distributional approaches are two major paradigms to classify a term pair into a fixed inventory of lexical relations, or to predict it as random (meaning the two terms are un-related) (Shwartz and Dagan, 2016; Wang et al., 2017a). Path-based approaches use dependency paths connecting two terms to infer lexical relations (Washio and Kato, 2018a; Roller et al., 2018). The paths usually describe relations between terms explicitly, but require the two terms co-occur in a sentence, leading to the “low coverage” problem. Apart from Hearst patterns (Hearst, 1992), there are few high-quality textual patterns to recognize lexical relations other than hypernymy. Distributional approaches consider the global contexts of terms to predict lexical relations using word embeddings (Baroni et al., 2012; Glavas and Vulic, 2018). They are reported to outperform several path-based approaches, but can suffer from the “lexical memorization” problem (Levy et al., 2015; Shwartz and Dagan, 2016). This is because some supervised distributional approaches learn properties of two terms separately, instead of how two terms relate to each other in the embedding space.

In this paper, we aim at improving distributional approaches by learning lexical relation representations in hyperspherical embedding space, named hyperSpherical Relation Embeddings (SphereRE). Consider the example w.r.t. the term car and three types of lexical relations.

Figure 1: An example of hyperspherical learning w.r.t. the term car and three types of lexical relations.
guish the three types of lexical relations well (i.e., hypernymy, synonymy and meronymy). Instead of learning individual term embeddings, we directly map all the relation triples to the hyperspherical embedding space such that different types of lexical relations have diverse embeddings in terms of angles. For example, the angle between embeddings of \((\text{car, hypernymy, vehicle})\) and \((\text{car, synonymy, auto})\) is large. In contrast, that of \((\text{car, synonymy, automobile})\) and \((\text{car, synonymy, auto})\) is small. As a result, different types of lexical relations can be distinguished. Moreover, by learning representations of lexical relation triples explicitly, our work addresses “lexical memorization” (Levy et al., 2015) from a distributional aspect.

To learn SphereRE vectors for lexical relation triples, we minimize embedding distances of term pairs that are likely to share the same lexical relation in both labeled and unlabeled data, and maximize embedding distances of different lexical relations. The distances in the hyperspherical space are defined based on the angles of embeddings. In this work, we first propose a relation-aware semantic projection model to estimate probabilistic distributions of lexical relations over unlabeled data. The SphereRE vectors are efficiently learned by Monte-Carlo techniques by transductive learning. Finally, a neural network based classifier is trained using all the features to make the final predictions of lexical relations over all unlabeled data.

We evaluate SphereRE over four benchmark datasets and the CogALex-V shared task (Santus et al., 2016a), and confirm that SphereRE is highly effective, outperforming state-of-the-art. We also evaluate the embedding quality of SphereRE.

The rest of this paper is organized as follows. Section 2 summarizes the related work. We present SphereRE in Section 3. Experiments are illustrated in Section 4, with the conclusion shown in Section 5.

2 Related Work

We briefly overview related work on lexical relation classification and hyperspherical learning.

2.1 Lexical Relation Classification

Among all methods, path-based and distributional approaches are two major paradigms (Shwartz and Dagan, 2016). For hypernymy relations, Hearst patterns (Hearst, 1992) are lexical patterns frequently employed, summarized in Wang et al. (2017a). Shwartz et al. (2016) employ an LSTM-based neural network to learn representations of dependency paths. Roller et al. (2018) use Hearst pattern based statistics derived from a large text corpus to detect hypernymy relations. For other lexical relations, LexNET (Shwartz and Dagan, 2016) extends Shwartz et al. (2016) to classify multiple types of lexical relations based on an integrated neural network. This type of methods requires that the two terms co-occur in a sentence. Washio and Kato (2018a) address the “low coverage” issue by augmenting dependency paths. Distributional approaches employ term representations to predict lexical relations, which exploit global contexts of terms. Traditional methods use a combination of two terms’ embeddings as the representation, such as vector concatenation (Baroni et al., 2012; Roller and Erk, 2016), vector difference (Roller et al., 2014; Weeds et al., 2014; Vylomova et al., 2016), etc. After that, a classifier is trained to predict lexical relations. Although distributional methods do not require the co-occurrence of two terms, they suffer from “lexical memorization” (Levy et al., 2015). It means the algorithms only learn the properties of the two terms, rather than the relations between them. Recently, more complicated neural networks have been proposed. Glavas and Vulic (2018) propose a Specialization Tensor Model to discriminate between four lexical relations. The model learns different specializations of input distributional embeddings w.r.t. term pairs in order to predict different types of lexical relations. Attia et al. (2016) employ a convolutional neural network in a multi-task setting. Nguyen et al. (2016, 2017b) distinguish antonymy and synonymy via word embeddings and path-based neural networks. Similar research is presented in Hashimoto et al. (2015); Washio and Kato (2018b); Chen et al. (2018); Bouraoui et al. (2018). A few works learn relation embeddings for other NLP applications (Jameel et al., 2018; Joshi et al., 2018).

Another research direction is to learn specializing embeddings. Yu et al. (2015); Luu et al. (2016); Nguyen et al. (2017a); Vulic and Mrksic (2018) (and a few others) learn hypernymy embeddings considering hierarchical structure of hypernymy relations. For other lexical relations, Mrksic et al. (2017) present the model Attract-Repel to improve qualities of word embeddings for synonymy recognition. However, they focus on one
particular lexical relation, not capable of distinguishing multiple types of lexical relations.

2.2 Hyperspherical Learning

The work of hyperspherical learning is mostly in computer vision. Liu et al. (2017) propose a hyperspherical network (SphereNet) for image classification. It learns angular representations on hyperspheres using hyperspherical convolution units. Wang et al. (2017c) apply the $L_2$ hypersphere embedding technique to face verification, optimizing cosine similarity for feature normalization. In NLP, hyperspherical learning has not been extensively used. Masumura et al. (2017) introduce hyperspherical clustering for document categorization. Mei and Wang (2016) leverage hyperspherical representations as knowledge graph embeddings. To our knowledge, few methods employ hyperspherical learning to learn representations for NLP applications. In our work, we focus on lexical relation classification and present the SphereRE model to address this problem.

3 The SphereRE Model

We introduce the Hyperspherical Relation Embedding (SphereRE) model in detail.

3.1 Learning Objective

We start with some basic notations. Let $D$ and $U$ be the labeled and unlabeled sets, consisting of term pairs $(x_i, y_i)$. Each pair $(x_i, y_i)$ corresponds to a pre-defined lexical relation type $r_i \in R$. The task of our work is to predict the lexical relation type $r_i$ for each pair $(x_i, y_i) \in U$ based on $D$.

Denote $\vec{x}_i$ (or $\vec{y}_i$) as the embedding of word $x_i$ (or $y_i$), pre-trained using any neural language models. For each lexical relation type $r_m \in R$, we learn a mapping function $f_m(\vec{x}_i)$ that maps the relation subject $x_i$ to the relation object $y_i$ in the embedding space if $x_i$ and $y_i$ have the lexical relation type $r_m$. Hence, we aim at minimizing the objective function $J_f$ with $I(\cdot)$ as the indicator function:

$$J_f = \sum_{i=1}^{\left| D \right|} \sum_{r_m \in R} I(r_i = r_m) \| f_m(\vec{x}_i) - \vec{y}_i \|^2$$

To represent lexical relation triples in the (original) embedding space, we utilize the vector difference model (Roller et al., 2014; Weeds et al., 2014; Vylomova et al., 2016). Combining the model $J_f$, given a term pair $(x_i, y_i)$ with lexical relation type $r_i$, the representation of the triple is $f_i(\vec{x}_i) - \vec{x}_i$.

Next, we consider the hyperspherical learning objective. Based on the assumption in Figure 1, we define a symmetric function $g(\cdot, \cdot)$ to quantify the distance between two representations of lexical relation triples in the SphereRE space. Following Roller et al. (2014); Weeds et al. (2014); Vylomova et al. (2016), we employ the vector difference model to represent the relation embedding of a term pair. Because we aim at learning representations for both labeled and unlabeled data in order to make predictions, for two pairs $(x_i, y_i)$ and $(x_j, y_j)$ with lexical relation types $r_i$ and $r_j ((x_i, y_i), (x_j, y_j) \in D \cup U)$, we minimize the following function:

$$\delta(r_i, r_j) g(f_i(\vec{x}_i) - \vec{x}_i, f_j(\vec{x}_j) - \vec{x}_j)$$

where $\delta(r_i, r_j)$ is the sign function that returns 1 if the two pairs share the same lexical relation type (i.e., $r_i = r_j$) and -1 otherwise. Hence, embedding distances of term pairs that share the same lexical relation type are minimized. Embedding distances of term pairs with different lexical relation types are maximized. Refer to Figure 2 for a geometric interpretation of the objective.

![Figure 2: A geometric interpretation of hyperspherical learning. “v(car, auto)” is the embedding vector w.r.t. the term pair “(auto, car)” (i.e., $f_i(\vec{x}_i) - \vec{x}_i$) based on vector difference and $J_f$, characterizing the lexical relation of the two terms in the original embedding space. For simplicity, we use an arrow to represent the embedding $f_i(\vec{x}_i) - \vec{x}_i$.](image-url)

In summary, the objective function of lexical relation representation learning in the hyperspherical embedding space $J_g$ is defined as follows:

$$J_g = \sum_{i,j} \delta(r_i, r_j) g(f_i(\vec{x}_i) - \vec{x}_i, f_j(\vec{x}_j) - \vec{x}_j)$$
Let $\Theta$ be all parameters in the model. The general objective of SphereRE is defined as follows, with $\lambda_1$ and $\lambda_2$ as balancing hyperparameters:

$$J(\Theta) = J_f + \lambda_1 J_g + \lambda_2 \|\Theta\|^2$$

It is computationally intractable to minimize $J(\Theta)$. The reasons are twofold: i) The lexical relation types $r_i$ of all pairs $(x_i, y_i) \in U$ should be predicted before we can minimize $J(\Theta)$. ii) The definition of $J_g$ does not directly determine how to generate the representations of lexical relation triples. Additionally, minimizing $J(\Theta)$ requires the traversal of $D$ and $U$ in quadratic time, leading to the high computational cost.

In the following, we present a relation-aware semantic projection model as the function $f_m(\cdot)$. It is employed to approximate $r_i$ for all $(x_i, y_i) \in U)$. Next, the representation learning process of lexical relation triples and the lexical relation classification algorithms are introduced in detail.

### 3.2 Relation-aware Semantic Projection

For each pair $(x_i, y_i) \in U$, we approximate $r_i$ from a probabilistic perspective, as an initial prediction step. Following Wang and He (2016); Yamane et al. (2016); Wang et al. (2017b), for each lexical relation type $r_m \in R$, we utilize a mapping matrix $M_m \in \mathbb{R}^{d \times d}$ as $f_m(x_i)$ where $d$ is the dimension of pre-trained word embeddings. After adding a Tikhonov regularizer on $M_m$, the learning objective function $J_m$ w.r.t. one specific lexical relation type $r_m \in R$ over $D$ can be re-written as follows:

$$J_m = \sum_{i=1}^{\lvert D \rvert} I(r_i = r_m) \|M_m x_i - y_i\|^2 + \mu \|M_m\|^2_F$$

Therefore, $J_f = \sum_{r_m \in R} J_m$. The minimization of $J_m$ has a closed-form solution. The optimal solution $M_m^*$ is as follows:

$$M_m^* = \arg \min_{M_m} J_m = (X_m^T X_m + \mu I)^{-1} X_m^T Y_m$$

(1)

where $X_m$ and $Y_m$ are two $n_m \times d$ data matrices, with $n_m$ being the number of term pairs that have the lexical relation type $r_m \in R$ in $D$. The $i$-th rows of $X_m$ and $Y_m$ are the embedding vectors of the $i$-th sample $(x_i, y_i) \in D$ that has the lexical relation type $r_m \in R$. $E$ is a $d \times d$ identity matrix.

For each lexical relation type $r_m \in R$, we train a semantic projection model based on Eq. (1). After that, a simple lexical relation prediction classifier is trained over $D$ based on the following $|R| \times d$-dimensional feature vector $F(x_i, y_i)$:

$$F(x_i, y_i) = (M_1 x_i - y_i) \oplus \cdots \oplus (M_{|R|} x_i - y_i)$$

where $\oplus$ is the vector concatenation operator. $M_1, \cdots, M_{|R|}$ are projection matrices w.r.t. $|R|$ lexical relation types $r_1, \cdots, r_{|R|}$.

Based on $J_m$, if $(x_i, y_i)$ has the lexical relation type $r_m$, the norm of $M_m x_i - y_i$ is likely to be small. On the contrary, the norms of $M_m x_i - y_i$ $(1 \leq n \leq |R|, n \neq m)$ are likely to be large. Therefore, the features are highly discriminative for lexical relation classification.

For each pair $(x_i, y_i) \in U$, the classifier outputs an $|R|$-dimensional probabilistic distribution over all lexical relation types $R$. In this work, we denote $p_{i,m}$ as the probability of $(x_i, y_i) \in U$ having the lexical relation type $r_m \in R$.

### 3.3 Relation Representation Learning

After we have computed the probability $p_{i,m}$ for all $(x_i, y_i) \in U$ and all $r_m \in R$, we focus on the objective $J_g$. The goal is to learn a $d_r$-dimensional vector $\vec{r}_i$ for each $(x_i, y_i) \in D \cup U$, regarded the representation of the lexical relation triple (named the SphereRE vector).

To avoid the high complexity and the propagation effect of predicted errors, inspired by Perozzi et al. (2014); Grover and Leskovec (2016), we reformulate $J_g$ and the function $g(\cdot, \cdot)$ via the Skip-gram model (Mikolov et al., 2013a) over neighboring graphs. Let $Nb(x_i, y_i)$ be the neighbors of a term pair $(x_i, y_i)$ in the SphereRE space, where each term pair $(x_j, y_j) \in Nb(x_i, y_i)$ is likely to share the same lexical relation type as $(x_i, y_i)$. To ensure that term pairs with the same lexical relation type have similar SphereRE vectors, the problem of optimizing $J_g$ can be reformulated by maximizing the probability of predicting the neighbors of $(x_i, y_i)$ given its SphereRE vector $\vec{r}_i$. Therefore, we define a new objective function $J_g$ to re-
unknown, we compute the weight because the lexical relation types of both pairs are all possible lexical relation types in $R$.

A remaining problem is to define the neighborhood $Nh(x_i, y_i)$ properly, to preserve the hyperspherical similarity property of the distance function $g(f_i(\vec{x}) - \vec{x}_i, f_j(\vec{x}) - \vec{x}_j)$. In this work, we introduce a weight factor $w_{i,j}$ in terms of angles is defined using the weighted cosine similarity in the range of $(0, 1)$:

$$w_{i,j} = \frac{1}{2} p_{j,m}(\cos(M_m \vec{x}_i - \vec{x}_i, M_m \vec{x}_j - \vec{x}_j) + 1)$$

A similar case holds for $(x_i, y_i) \in U$ and $(x_j, y_j) \in D$. If $(x_i, y_i) \in U$ and $(x_j, y_j) \in U$, because the lexical relation types of both pairs are unknown, we compute the weight $w_{i,j}$ by summing up all the weighted cosine similarities over all possible lexical relation types in $R$:

$$w_{i,j} = \frac{1}{2} \sum_{r_{m} \in R} p_{l,m}p_{j,m'}(\cos(M_m \vec{x}_i - \vec{x}_i, M_m \vec{x}_j - \vec{x}_j) + 1)$$

Readers can also refer to Table 1 for a summarization of the choices of $w_{i,j}$.

To reduce computational complexity, we propose a Monte-Carlo based sampling and learning method to learn SphereRE vectors based on the values of $w_{i,j}$. The algorithm is illustrated in Algorithm 1. It starts with the random initialization of SphereRE vector $\vec{r}_i$ for each $(x_i, y_i) \in D \cup U$. An iterative process randomly selects one pair $(x_i, y_i)$ as the starting point. The next pair $(x_j, y_j)$ is selected with probability as follows:

$$Pr((x_j, y_j)|(x_i, y_i)) = \frac{w_{i,j}}{\sum_{(x_j', y_j') \in D_{\text{mini}}(x_i, y_i)} w_{i,j'}}$$

where $D_{\text{mini}}$ is a mini-batch of term pairs randomly selected from $D \cup U$. In this way, the algorithm only needs to traverse $|D_{\text{mini}}|$ pairs instead of $|D| + |U|$ pairs. This process continues, resulting in a sequence of pairs, denoted as $S$: $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{|S|}, y_{|S|})\}$. Denote $l$ as the window size. We approximate $J'$ in Eq. (2) by

$$-\sum_{(x_i, y_i) \in S} \sum_{j=i+1}^{i+l} \log Pr((x_j, y_j)|\vec{r}_i)$$

using the negative sampling training technique of the Skip-gram model (Mikolov et al., 2013a,b).

The values of SphereRE vectors $\vec{r}_i$ are continuously updated until all the iterations stop. We can see that $\vec{r}_i$s are the low-dimensional representations of lexical relation triples, encoded in the hyperspherical space. The process is shown in Algorithm 1.

### Algorithm 1 SphereRE Learning

1. for each $(x_i, y_i) \in D \cup U$ do
2. Randomly initialize SphereRE vector $\vec{r}_i$;
3. end for
4. for $i = 1$ to max iteration do
5. Sample a sequence based on Eq. (3):
6. Update all SphereRE vectors $\vec{r}_i$ by minimizing
7. end for

In practice, we find that there is a drawback of the sampling process. Because the predictions for all $(x_i, y_i) \in U$ are probabilistic, it leads to the situation where the algorithm prefers to choose term pairs in $D$ to form the sequence $S$. The low sampling rate of $U$ results in the poor representation learning quality of these pairs. Here, we employ a boosting approach to increase chances.
of \((x_i, y_i) \in U\) being selected based on stratified sampling. The values of all probabilities \(p_{i,m}\) are multiplied by a factor \(\gamma > 1\), i.e., \(p_{i,m} \leftarrow p_{i,m} \gamma\).  

3.4 Lexical Relation Classification

Finally, we train a lexical relation classifier. For each pair \((x_i, y_i) \in D\), we train a classifier over \([|R| \times d + d_r]\)-dimensional feature set \(F^*(x_i, y_i)\):

\[
F^*(x_i, y_i) = F(x_i, y_i) \oplus \vec{r}_i
\]

where \(F(x_i, y_i)\) are \(|R| \times d\)-dimensional projection-based features. \(\vec{r}_i\) is the SphereRE vector of \((x_i, y_i)\) that encodes the relation triple in the SphereRE space.

We follow the work (Shwartz and Dagan, 2016) by using a fully-connected feed-forward neural network, shown in Figure 3. The input layer has \(|R| \times d + d_r\) nodes. We add only one hidden layer, followed by an \(|R|\)-dimensional output layer with softmax as the prediction function. The neural network is trained using the stochastic gradient descent algorithm, and is employed to predict the lexical relations for all \((x_i, y_i) \in U\). The high-level procedure is summarized in Algorithm 2.

![Diagram of the neural network architecture.](#)

**Algorithm 2: Lexical Relation Classification**

1: for each lexical relation type \(r_m \in R\) do
2: \hspace{1em} Compute \(M_m^d\) by Eq. (1);
3: \hspace{1em} end for
4: Train a classifier over \(D\) over \(F(x_i, y_i)\);
5: for each pair \((x_i, y_i) \in U\) do
6: \hspace{1em} Predict distribution \(p_{i,m}\) by the classifier;
7: \hspace{1em} end for
8: Learning \(\vec{r}_i\) for all \((x_i, y_i) \in D \cup U\) by Algorithm 1;
9: Train a neural network over \(D\) by features \(F^*(x_i, y_i)\);
10: for each pair \((x_i, y_i) \in U\) do
11: \hspace{1em} Predict the lexical relation \(r_i\) by the neural network;
12: \hspace{1em} end for

3Note that although we do not explicitly optimize \(J_0\) or construct the SphereRE space directly, the SphereRE vectors learned by Algorithm 1 (i.e., \(\vec{r}_i\)) reflect the clear distinctions of triples with different lexical relation types. Further analysis of SphereRE vectors will be shown in experiments.

4 Experiments

In this section, we conduct extensive experiments to evaluate SphereRE and compare it with state-of-the-art to make the convincing conclusion.

4.1 Datasets and Experimental Settings

In the experiments, we train a fastText model (Bojanowski et al., 2017) over the English Wikipedia corpus to generate term embeddings. The dimensionality \(d\) is set to 300. To evaluate the effectiveness of SphereRE, we use four public datasets for multi-way classification of lexical relations: K&H+N (Necsulescu et al., 2015), BLESS (Baroni and Lenci, 2011), ROOT09 (Santus et al., 2016b) and EVALution (Santus et al., 2015). We also evaluate SphereRE over the subtask 2 of the CogALex-V shared task (Santus et al., 2016a). The statistics are summarized in Table 2.

We follow the exact same experimental settings to partition the four public datasets into training, validation and testing sets as in (Shwartz and Dagan, 2016). The partition of the CogALex dataset is the same as those in the default settings of the CogALex-V shared task (Santus et al., 2016a). The default settings for SphereRE are as follows: \(\mu = 0.001\), \(d_r = 300\), \(|D_{min}| = 20\), \(|S| = 100\), \(\gamma = 2\) and \(l = 3\). We run Algorithm 1 in 500 iterations. We also report how the changes of the neural network architecture and parameters affect the performance over the validation sets afterwards. It should be further noted that we do not set the values of \(\lambda_1\) and \(\lambda_2\) in the implementation because we employ sampling based techniques to learn \(\vec{r}_i\), instead of directly optimizing \(J(\Theta)\).

4.2 Experiments over Four Public Datasets

We report the results of SphereRE and compare it with state-of-the-art over four public datasets.

4.2.1 General Performance

To compare SphereRE with others, we consider following baselines:

- **Concat** (Baroni et al., 2012), **Diff** (Weeds et al., 2014): They are classical distributional methods using vector concatenation and vector difference as features. A neural network without hidden layers is trained.
- **NPB** (Shwartz et al., 2016): It uses a path-based LSTM neural network to classify lexical relations. It is implemented by Shwartz.
and Dagan (2016) and only considers dependency paths.

- LexNET (Shwartz and Dagan, 2016): It is built upon Shwartz et al. (2016), which combines representations of dependency paths and word embeddings for classification.
- Concat_h, Diff_h, LexNET_h: They are variants of Concat, Diff and LexNET, with one hidden layer between the input and the output layer.
- NPB+Aug, LexNET+Aug (Washio and Kato, 2018a): They are variants of NPB and LexNET. The dependency paths used in the two original systems have been augmented in order to improve the pattern coverage.

The results of SphereRE and all the baselines are summarized in Table 3. We compute the Precision, Recall and F1 score for each lexical relation, and report the average scores over all the relations, weighted by the support. We can see that classification distributional approaches perform worse than integrated neural networks (such as Shwartz and Dagan (2016)), because they are not capable of learning the true relations between terms. The proposed approach SphereRE consistently outperforms all the baselines over the four datasets in terms of F1 scores. When the type of lexical relations becomes larger (e.g., EVALution), the improvement of SphereRE are less significant than that of other datasets (e.g., BLESS, ROOT09). The most possible cause is that errors induced by relation-aware semantic projection are more likely to propagate to subsequent steps.

### 4.2.2 Study on Neural Network Architectures

We adjust the neural network architecture (shown in Figure 3) and report the performance over the validation sets in Figure 4. As shown, adding more hidden layers does not improve the performance of lexical relation classification. In some datasets (e.g., EVALution), the performance even drops, indicating a sign of overfitting. We change the number of hidden nodes when we use one hidden layer in the network. The results show that the setting does not affect the performance greatly.

![Figure 4: Network structure analysis.](image)

### 4.2.3 Study on Monte-Carlo Sampling

We continue to study how the settings of Monte-Carlo sampling affect the quality of the SphereRE vectors. We adjust the number of iterations and the parameter $\gamma$. The performance is shown in Figure 5. As seen, more iterations contribute to the higher quality of embeddings. After a sufficient number of iterations (> 500), the performance becomes stable. As for the choice of $\gamma$, smaller values lead to the low sampling rates of unlabeled data, hence lower the prediction performance. In contrast, an overly large $\gamma$ induces too many errors in relation-aware semantic projection to the sampling process. Hence, a balanced setting of $\gamma$ is required.

![Figure 5: MC sampling analysis.](image)
#### Table 3: Performance comparison of lexical relation classification over four public datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concat</td>
<td>0.909</td>
<td>0.906</td>
<td>0.904</td>
<td>0.811</td>
<td>0.812</td>
<td>0.811</td>
<td>0.636</td>
<td>0.675</td>
<td>0.646</td>
<td>0.531</td>
<td>0.544</td>
<td>0.525</td>
<td></td>
</tr>
<tr>
<td>ConcatA</td>
<td>0.888</td>
<td>0.886</td>
<td>0.885</td>
<td>0.801</td>
<td>0.803</td>
<td>0.802</td>
<td>0.627</td>
<td>0.655</td>
<td>0.638</td>
<td>0.521</td>
<td>0.531</td>
<td>0.528</td>
<td></td>
</tr>
<tr>
<td>Diff</td>
<td>0.941</td>
<td>0.942</td>
<td>0.941</td>
<td>0.861</td>
<td>0.859</td>
<td>0.860</td>
<td>0.683</td>
<td>0.692</td>
<td>0.686</td>
<td>0.536</td>
<td>0.54</td>
<td>0.539</td>
<td></td>
</tr>
<tr>
<td>DiffA</td>
<td>0.713</td>
<td>0.604</td>
<td>0.55</td>
<td>0.759</td>
<td>0.756</td>
<td>0.755</td>
<td>0.788</td>
<td>0.789</td>
<td>0.788</td>
<td>0.53</td>
<td>0.537</td>
<td>0.503</td>
<td></td>
</tr>
<tr>
<td>NPB</td>
<td>0.985</td>
<td>0.985</td>
<td>0.984</td>
<td>0.895</td>
<td>0.893</td>
<td>0.893</td>
<td>0.812</td>
<td>0.814</td>
<td>0.813</td>
<td>0.601</td>
<td>0.607</td>
<td>0.606</td>
<td></td>
</tr>
<tr>
<td>LexNET</td>
<td>0.984</td>
<td>0.985</td>
<td>0.984</td>
<td>0.895</td>
<td>0.892</td>
<td>0.893</td>
<td>0.812</td>
<td>0.814</td>
<td>0.813</td>
<td>0.589</td>
<td>0.587</td>
<td>0.583</td>
<td></td>
</tr>
<tr>
<td>NPB+Aug</td>
<td>-</td>
<td>-</td>
<td>0.897</td>
<td>-</td>
<td>-</td>
<td>0.842</td>
<td>-</td>
<td>-</td>
<td>0.778</td>
<td>-</td>
<td>-</td>
<td>0.489</td>
<td></td>
</tr>
<tr>
<td>LexNET+Aug</td>
<td>-</td>
<td>-</td>
<td>0.975</td>
<td>-</td>
<td>-</td>
<td>0.927</td>
<td>-</td>
<td>-</td>
<td>0.906</td>
<td>-</td>
<td>-</td>
<td>0.545</td>
<td></td>
</tr>
</tbody>
</table>

#### Table 4: Feature analysis in terms of F1 score.

<table>
<thead>
<tr>
<th>Features</th>
<th>Dataset</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o. SphereRE vectors</td>
<td>0.998</td>
<td>0.918</td>
<td>0.82</td>
<td>0.384</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w. SphereRE vectors</td>
<td>0.990</td>
<td>0.938</td>
<td>0.861</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Improvement</td>
<td>+2.2%</td>
<td>+9.0%</td>
<td>+11.1%</td>
<td>+3.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 5: Performance comparison over the CogALex-V shared task. (Due to space limitation, we only list the performance of top systems in CogALex-V.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Relation</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Pre</th>
<th>Rec</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attia et al. (2016)</td>
<td>SYN</td>
<td>0.204</td>
<td>0.448</td>
<td>0.347</td>
<td>0.457</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shwartz and Dagan (2016)</td>
<td>ANT</td>
<td>0.297</td>
<td>0.526</td>
<td>0.493</td>
<td>0.445</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glavas and Vulic (2018)</td>
<td>otp</td>
<td>0.221</td>
<td>0.504</td>
<td>0.498</td>
<td>0.453</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SphereRE</td>
<td>-</td>
<td>0.256</td>
<td>0.519</td>
<td>0.533</td>
<td>0.471</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 4.2.4 Feature Analysis

We further study whether adding the SphereRE vectors contributes to lexical relation classification. We remove all the these embeddings and use the rest of the features to make prediction based on the same neural architecture and parameter settings. The results are shown in Table 4. By learning the SphereRE vectors and adding them to the classifier, the performance improves in all four datasets.

#### 4.3 Experiments over the CogALex-V Shared Task

We evaluate SphereRE over the CogALex-V shared task (Santus et al., 2016a), where participants are asked to classify 4,260 term pairs into 5 lexical relations: synonymy, antonymy, hypernymy, meronymy and random. The training set contains 3,054 pairs. This task is the most challenging because i) it considers random relations as noise, discarding it from the averaged F1 score; ii) the training set is small; and iii) it enforces lexical split of the training and testing sets, disabling “lexical memorization” (Levy et al., 2015).

In this shared task, GHHH (Attia et al., 2016) and LexNET (Shwartz and Dagan, 2016) are top-two systems with the highest performance. The most recent work on CogALex-V is STM (Glavas and Vulic, 2018). SphereRE achieves the averaged F1 score of 47.1% (excluding the random relations), outperforming state-of-the-art. Additionally, as reported in previous studies, the “lexical memorization” effect (Levy et al., 2015) is rather severe for hypernymy relations. Although SphereRE is fully distributional, it achieves the highest F1 score of 53.8%.

#### 4.4 Analysis of SphereRE Vector Qualities

We conduct additional experiments to evaluate the qualities of Sphere vectors. The first set of experiments evaluates whether top-$k$ most similar relation triples of a given relation triple share the same lexical relation type. This task is called top-$k$ similar lexical relation retrieval. In this task, the similarity between two relation triples is quantified by the cosine similarity of the two corresponding SphereRE vectors. The score is reported by Precision@$k$. Higher Precision@$k$ scores indicate SphereRE vectors with better quality, because lexical relation triples with the same lexical relation type should have similar Sphere vectors. In the experiments, we compute the Precision@$k$ over all the labeled (training) and unlabeled (testing) sets of all five datasets. The results are shown in Table 6 in terms of Average Precision@$k$ (AP@$k$) (with $k = 1, 5, 10$).

As seen, SphereRE has near perfect performance (over 95% for AP@1, over 90% for AP@5 and AP@10) over training sets of all five datasets. This is because in representation learning, all the labels (i.e., lexical relation types) of these term pairs are already known. Hence, SphereRE preserves distributional characteristics of these labeled datasets well. As for unlabeled datasets, the performance drops slightly over K&H+N, BLESS and ROOT09. The performance is not very satis-
4.5 Error Analysis

For error analysis, we randomly sample 300 cases of prediction errors and ask human annotators to analyze the most frequent causes. We present several cases in Table 7. The largest number of errors (approximately 42%) occur due to the random relations in K&H+N, BLESS, ROOT09 and CogALex. These relations are large in quantity and blurry in semantics, misleading the classifier to predict other lexical relations as random.

Another large proportion of errors (about 31%) are related to unbalanced ratio of relations (apart from random). The number of some types of lexical relation triples in the training set is small (e.g., Meronym in EVALution, Synonym in CogALex). As a result, the representation learning w.r.t. these relation triples is relatively of lower quality.

5 Conclusion and Future Work

In this paper, we present a representation learning model to distinguish lexical relations based on Hyperspherical Relation Embeddings (SphereRE). It learns representations of lexical relation triples by mapping them to the hyperspherical embedding space. The lexical relations between term pairs are predicted using neural networks over the learned embeddings. Experiments over four benchmark datasets and CogALex-V show SphereRE outperforms state-of-the-art methods.

In the future, we will improve our model to deal with datasets containing a relatively large number of lexical relation types and random term pairs. Additionally, the mapping technique used for relation-aware semantic projection can be further improved to model different linguistic properties of lexical relations (e.g., the “one-to-many” mappings for meronymy).

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904.
References


Goran Glavas and Ivan Vulic. 2018. Discriminating between lexico-semantic relations with the specialization tensor model. In NAACL.


Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *SEM.


Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In ACL.
