Embedding Semantic Relations into Word Representations

Danushka Bollegala University of Liverpool **Takanori Maehara** Shizuoka University **Ken-ichi Kawarabayashi** National Institute of Informatics
JST, ERATO, Kawarabayashi Large Graph Project.

Abstract

Learning representations for semantic relations is important for various tasks such as analogy detection, relational search, and relation classification. Although there have been several proposals for learning representations for individual words, learning word representations that explicitly capture the semantic relations between words remains under developed. We propose an unsupervised method for learning vector representations for words such that the learnt representations are sensitive to the semantic relations that exist between two words. First, we extract lexical patterns from the co-occurrence contexts of two words in a corpus to represent the semantic relations that exist between those two words. Second, we represent a lexical pattern as the weighted sum of the representations of the words that co-occur with that lexical pattern. Third, we train a binary classifier to detect relationally similar vs. non-similar lexical pattern pairs. The proposed method is unsupervised in the sense that the lexical pattern pairs we use as train data are automatically sampled from a corpus, without requiring any manual intervention. Our proposed method statistically significantly outperforms the current state-of-the-art word representations on three benchmark datasets for proportional analogy detection, demonstrating its ability to accurately capture the semantic relations among words.

1 Introduction

Representing the semantics of words and relations are fundamental tasks in Knowledge Representation (KR). Numerous methods for learning distributed word representations have been proposed in the NLP community [Turian *et al.*, 2010; Collobert *et al.*, 2011; Mikolov *et al.*, 2013b; 2013a; Pennington *et al.*, 2014]. Distributed word representations have shown to improve performance in a wide-range of tasks such as, machine translation [Cho *et al.*, 2014], semantic similarity measurement [Mikolov *et al.*, 2013d; Pennington *et al.*, 2014], and word sense disambiguation [Huang *et al.*, 2012].

Despite the impressive performance of representation learning methods for individual words, existing methods use only co-occurrences between words, ignoring the rich semantic relational structure. The context in which two words co-occur provides useful insights into the semantic relations that exist between those two words. For example, the sentence ***ostrich*** *is a large* ***bird*** not only provides the information that *ostrich* and *bird* are co-occurring, but also describes how they are related via the lexical pattern **X** *is a large* **Y**, where slot variables **X** and **Y** correspond to the two words between which the relation holds. If we can somehow embed the information about the semantic relations R that are associated with a particular word w into the representation of w, then we can construct richer semantic representation than the pure co-occurrence-based word representations. Although the word representations learnt by co-occurrence prediction methods [Mikolov *et al.*, 2013d; Pennington *et al.*, 2014] have implicitly captured a certain degree of relational structure, it remains unknown how to explicitly embed the information about semantic relations into word representations.

We propose a method for learning word representations that explicitly encode the information about the semantic relations that exist between words. Given a large corpus, we extract lexical patterns that correspond to numerous semantic relations that exist between word-pairs (x_i, x_j). Next, we represent each word x_i in the vocabulary by a d-dimensional vector $\boldsymbol{x}_i \in \mathbb{R}^d$. Word representations can be initialized either randomly or by using pre-trained word representations. Next, we represent a pattern p by the weighted average of the vector differences $(\boldsymbol{x}_i - \boldsymbol{x}_j)$ corresponding to word-pairs (x_i, x_j) that co-occur with p in the corpus. This enables us to represent a pattern p by a d-dimensional vector $\boldsymbol{p} \in \mathbb{R}^d$ in the same embedding space as the words. Using vector difference between word representations to represent semantic relations is motivated by the observations in prior work on word representation learning [Mikolov *et al.*, 2013d; Pennington *et al.*, 2014] where, for example, the difference of vectors representing *king* and *queen* has shown to be similar to the difference of vectors representing *man* and *woman*.

We model the problem of embedding semantic relations into word representations as an analogy prediction task where, given two lexical patterns, we train a binary classifier that predicts whether they are relationally similar. Our pro-

posed method is unsupervised in the sense that both positive and negative training instances that we use for training are automatically selected from a corpus, without requiring any manual intervention. Specifically, pairs of lexical patterns that co-occur with the same set of word-pairs are selected as positive training instances, whereas negative training instances are randomly sampled from pairs of patterns with low relational similarities. Our proposed method alternates between two steps (Algorithm 1). In the first step, we construct pattern representations from current word representations. In the second step, we predict whether a given pair of patterns is relationally similar using the computed representations of patterns in the previous step. We update the word representations such that the prediction loss is minimized.

Direct evaluation of word representations is difficult because there is no agreed gold standard for semantic representation of words. Following prior work on representation learning, we evaluate the proposed method using the learnt word representations in an analogy detection task. For example, denoting the word representation for a word w by $\boldsymbol{v}(w)$, the vector $\boldsymbol{v}(\mathrm{king}) - \boldsymbol{v}(\mathrm{man}) + \boldsymbol{v}(\mathrm{woman})$ is required to be similar to $\boldsymbol{v}(\mathrm{queen})$, than all the other words in the vocabulary. Similarity between two vectors is computed by the cosine of the angle between the corresponding vectors. The accuracy obtained in the analogy detection task with a particular word representation method is considered as a measure of its accuracy. In our evaluations, we use three previously proposed benchmark datasets for word analogy detection: SAT analogy dataset [Turney, 2005], Google analogy dataset [Mikolov *et al.*, 2013c], and SemEval analogy dataset [Jurgens *et al.*, 2012]. The word representations produced by our proposed method statistically significantly outperform the current state-of-the-art word representation learning methods on all three benchmark datasets in an analogy detection task, demonstrating the accuracy of the proposed method for embedding semantic relations in word representations.

2 Related Work

Representing words using vectors (or tensors in general) is an essential task in text processing. For example, in distributional semantics [Baroni and Lenci, 2010], a word x is represented by a vector that contains other words that co-occur with x in a corpus. Numerous methods for selecting co-occurrence contexts (e.g. proximity-based windows, dependency relations), and word association measures (e.g. pointwise mutual information (PMI), log-likelihood ratio (LLR), local mutual information (LLR)) have been proposed [Turney and Pantel, 2010]. Despite the successful applications of co-occurrence counting-based distributional word representations, their high dimensionality and sparsity is often problematic when applied in NLP tasks. Consequently, further post-processing such as dimensionality reduction, and feature selection is often required when using distributional word representations.

On the other hand, distributed word representation learning methods model words as d-dimensional real vectors and learn those vector representations by applying them to solve an auxiliary task such as language modeling. The dimensionality d is fixed for all the words in the vocabulary and, unlike distributional word representations, is much smaller (e.g. $d \in [10, 1000]$ in practice) compared to the vocabulary size. A pioneering work on word representation learning is the neural network language model (NNLMs) [Bengio *et al.*, 2003], where word representations are learnt such that we can accurately predict the next word in a sentence using the word representations for the previous words. Using backpropagation, word vectors are updated such that the prediction error is minimized.

Although NNLMs learn word representations as a by-product, the main focus on language modeling is to predict the next word in a sentence given the previous words, and not on learning word representations that capture word semantics. Moreover, training multi-layer neural networks with large text corpora is often time consuming. To overcome those limitations, methods that specifically focus on learning word representations that capture word semantics using large text corpora have been proposed. Instead of using only the previous words in a sentence as in language modeling, these methods use *all* the words in a contextual window for the prediction task [Collobert *et al.*, 2011]. Methods that use one or no hidden layers are proposed to improve the scalability of the learning algorithms. For example, the skip-gram model [Mikolov *et al.*, 2013c] predicts the words c that appear in the local context of a word x, whereas the continuous bag-of-words model (CBOW) predicts a word x conditioned on all the words c that appear in x's local context [Mikolov *et al.*, 2013a]. However, methods that use global co-occurrences in the entire corpus to learn word representations have shown to outperform methods that use only local co-occurrences [Huang *et al.*, 2012; Pennington *et al.*, 2014]. Word representations learnt using above-mentioned representation learning methods have shown superior performance over word representations constructed using the traditional counting-based methods [Baroni *et al.*, 2014].

Word representations can be further classified depending on whether they are task-specific or task-independent. For example, methods for learning word representations for specific tasks such as sentiment classification [Socher *et al.*, 2011], and semantic composition [Hashimoto *et al.*, 2014] have been proposed. These methods use label data for the target task to train supervised models, and learn word representations that optimize the performance on this target task. Whether the meaning of a word is task-specific or task-independent remains an interesting open question. Our proposal can be seen as a third alternative in the sense that we use task-independent pre-trained word representations as the input, and embed the knowledge related to the semantic relations into the word representations. However, unlike the existing task-specific word representation learning methods, we do not require manually labeled data for the target task (i.e. analogy detection).

3 Learning Word Representations

The local context in which two words co-occur provides useful information regarding the semantic relations that exist between those two words. For example, from the sentence ***Os-***

trich *is a large* ***bird*** *that primarily lives in Africa*, we can infer that the semantic relation IS-A-LARGE exists between *ostrich* and *bird*. Prior work on relational similarity measurement have successfully used such lexical patterns as features to represent the semantic relations that exist between two words [Duc *et al.*, 2010; 2011]. According to the *relational duality hypothesis* [Bollegala *et al.*, 2010], a semantic relation R can be expressed either *extensionally* by enumerating word-pairs for which R holds, or *intensionally* by stating lexico-syntactic patterns that define the properties of R.

Following these prior work, we extract lexical patterns from the co-occurring contexts of two words to represent the semantic relations between those two words. Specifically, we extract unigrams and bigrams of tokens as patterns from the *midfix* (i.e. the sequence of tokens that appear in between the given two words in a context) [Bollegala *et al.*, 2007b; 2007a]. Although we use lexical patterns as features for representing semantic relations in this work, our proposed method is not limited to lexical patterns, and can be used in principle with any type of features that represent relations. The strength of association between a word pair (u, v) and a pattern p is measured using the positive pointwise mutual information (PPMI), $f(p, u, v)$, which is defined as follows,

$$f(p, u, v) = \max(0, \log\left(\frac{g(p, u, v)g(*, *, *)}{g(p, *, *)g(*, u, v)}\right)). \quad (1)$$

Here, $g(p, u, v)$ denotes the number of co-occurrences between p and (u, v), and $*$ denotes the summation taken over all words (or patterns) corresponding to the slot variable. We represent a pattern p by the set $\mathcal{R}(p)$ of word-pairs (u, v) for which $f(p, u, v) > 0$. Formally, we define $\mathcal{R}(p)$ and its norm $|\mathcal{R}(p)|$ as follows,

$$\mathcal{R}(p) = \{(u, v)|f(p, u, v) > 0\} \quad (2)$$

$$|\mathcal{R}(p)| = \sum_{(u,v)\in\mathcal{R}(p)} f(p, u, v) \quad (3)$$

We represent a word $\boldsymbol{x}$ using a vector $\boldsymbol{x} \in \mathbb{R}^d$. The dimensionality of the representation, d, is a hyperparameter of the proposed method. Prior work on word representation learning have observed that the difference between the vectors that represent two words closely approximates the semantic relations that exist between those two words. For example, the vector $\boldsymbol{v}(\text{king}) - \boldsymbol{v}(\text{queen})$ has shown to be similar to the vector $\boldsymbol{v}(\text{man}) - \boldsymbol{v}(\text{woman})$. We use this property to represent a pattern p by a vector $\boldsymbol{p} \in \mathbb{R}^d$ as the weighted sum of differences between the two words in all word-pairs (u, v) that co-occur with p as follows,

$$\boldsymbol{p} = \frac{1}{|\mathcal{R}(p)|} \sum_{(u,v)\in\mathcal{R}(p)} f(p, u, v)(\boldsymbol{u} - \boldsymbol{v}). \quad (4)$$

For example, consider Fig. 1, where the two word-pairs (*lion*, *cat*), and (*ostrich*, *bird*) co-occur respectively with the two lexical patterns, p_1 = ***large Ys such as Xs***, and p_2 = ***X is a huge Y***. Assuming that there are no other co-occurrences between word-pairs and patterns in the corpus, the representations of the patterns p_1 and p_2 are given respectively by $\boldsymbol{p}_1 = \boldsymbol{x}_1 - \boldsymbol{x}_2$, and $\boldsymbol{p}_2 = \boldsymbol{x}_3 - \boldsymbol{x}_4$. We measure the

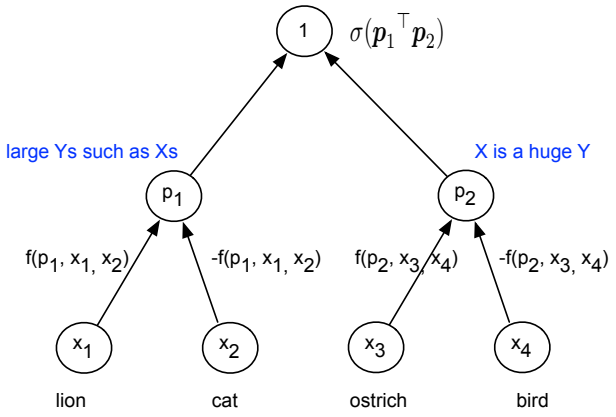

Figure 1: Computing the similarity between two patterns.

relational similarity between (x_1, x_2) and (x_3, x_4) using the inner-product $\boldsymbol{p}_1{}^\top \boldsymbol{p}_2$.

We model the problem of learning word representations as a binary classification task, where we learn representations for words such that they can be used to accurately predict whether a given pair of patterns are relationally similar. In our previous example, we would learn representations for the four words *lion*, *cat*, *ostrich*, and *bird* such that the similarity between the two patterns *large* ***Ys such as Xs***, and ***X is a huge Y*** is maximized. Later in Section 3.1, we propose an unsupervised method for selecting relationally similar (positive) and dissimilar (negative) pairs of patterns as training instances to train a binary classifier.

Let us denote the target label for two patterns p_1, p_2 by $t(p_1, p_2) \in \{1, 0\}$, where the value 1 indicates that p_1 and p_2 are relationally similar, and 0 otherwise. We compute the prediction loss for a pair of patterns (p_1, p_2) as the squared loss between the target and the predicted labels as follows,

$$L(t(p_1, p_2), p_1, p_2) = \frac{1}{2}(t(p_1, p_2) - \sigma(\boldsymbol{p}_1{}^\top \boldsymbol{p}_2))^2. \quad (5)$$

Different non-linear functions can be used as the prediction function $\sigma(\cdot)$ such as the logistic-sigmoid, hyperbolic tangent, or rectified linear units. In our preliminary experiments we found hyperbolic tangent, $\tanh$, given by

$$\sigma(\theta) = \tanh(\theta) = \frac{\exp(\theta) - \exp(-\theta)}{\exp(\theta) + \exp(-\theta)} \quad (6)$$

to work particularly well among those different non-linearities.

To derive the update rule for word representations, let us consider the derivative of the loss w.r.t. the word representation $\boldsymbol{x}$ of a word x,

$$\frac{\partial L}{\partial \boldsymbol{x}} = \frac{\partial L}{\partial \boldsymbol{p}_1}\frac{\partial \boldsymbol{p}_1}{\partial \boldsymbol{x}} + \frac{\partial L}{\partial \boldsymbol{p}_2}\frac{\partial \boldsymbol{p}_2}{\partial \boldsymbol{x}}, \quad (7)$$

where the partial derivative of the loss w.r.t. pattern representations are given by,

$$\frac{\partial L}{\partial \boldsymbol{p}_1} = \sigma'(\boldsymbol{p}_1{}^\top \boldsymbol{p}_2)(\sigma(\boldsymbol{p}_1{}^\top \boldsymbol{p}_2) - t(p_1, p_2))\boldsymbol{p}_2, \quad (8)$$

$$\frac{\partial L}{\partial \boldsymbol{p}_2} = \sigma'(\boldsymbol{p}_1{}^\top \boldsymbol{p}_2)(\sigma(\boldsymbol{p}_1{}^\top \boldsymbol{p}_2) - t(p_1, p_2))\boldsymbol{p}_1. \quad (9)$$

Algorithm 1 Learning word representations.

Input: Training pattern-pairs $\{(p_1^{(i)}, p_2^{(i)}, t(p_1^{(i)}, p_2^{(i)})\}_{i=1}^{N}$, dimensionality d of the word representations, and the maximum number of iterations T.

Output: Representation $\boldsymbol{x}_j \in \mathbb{R}^d$, of a word x_j for $j = 1, \ldots, M$, where M is the vocabulary size.

1: Initialize word vectors $\{\boldsymbol{x}_j\}_{j=1}^{M}$.
2: **for** $t = 1$ **to** T **do**
3: **for** $k = 1$ **to** K **do**
4: $\boldsymbol{p}_k = \frac{1}{|\mathcal{R}(p_k)|} \sum_{(u,v)\in\mathcal{R}(p_k)} f(p_k, u, v)(\boldsymbol{u} - \boldsymbol{v})$
5: **end for**
6: **for** $i = 1$ **to** N **do**
7: **for** $j = 1$ **to** M **do**
8: $\boldsymbol{x}_j = \boldsymbol{x}_j - \alpha_j^{(t)} \frac{\partial L}{\partial \boldsymbol{x}_j}$
9: **end for**
10: **end for**
11: **end for**
12: **return** $\{\boldsymbol{x}_j\}_{j=1}^{M}$.

Here, σ' denotes the first derivative of $\tanh$, which is given by $1 - \sigma(\theta)^2$. To simplify the notation we drop the arguments of the loss function.

From Eq. 4 we get,

$$\frac{\partial \boldsymbol{p_1}}{\partial \boldsymbol{x}} = \frac{1}{|\mathcal{R}(p_1)|} \left(h(p_1, u = x, v) - h(p_1, u, v = x)\right), \quad (10)$$

$$\frac{\partial \boldsymbol{p_2}}{\partial \boldsymbol{x}} = \frac{1}{|\mathcal{R}(p_2)|} \left(h(p_2, u = x, v) - h(p_2, u, v = x)\right), \quad (11)$$

where,

$$h(p, u = x, v) = \sum_{(x,v)\in\{(u,v)|(u,v)\in\mathcal{R}(p), u=x\}} f(p, x, v),$$

and

$$h(p, u, v = x) = \sum_{(u,x)\in\{(u,v)|(u,v)\in\mathcal{R}(p), v=x\}} f(p, u, x).$$

Substituting the partial derivatives given by Eqs. 8-11 in Eq. 7 we get,

$$\frac{\partial L}{\partial \boldsymbol{x}} = \lambda(p_1, p_2)[H(p_1, x) \sum_{(u,v)\in\mathcal{R}(p_2)} f(p_2, u, v)(\boldsymbol{u} - \boldsymbol{v}) + H(p_2, x) \sum_{(u,v)\in\mathcal{R}(p_1)} f(p_1, u, v)(\boldsymbol{u} - \boldsymbol{v})], \quad (12)$$

where $\lambda(p_1, p_2)$ is defined as

$$\lambda(p_1, p_2) = \frac{\sigma'({\boldsymbol{p}_1}^\top \boldsymbol{p}_2)(t(p_1, p_2) - \sigma({\boldsymbol{p}_1}^\top \boldsymbol{p}_2))}{|\mathcal{R}(p_1)||\mathcal{R}(p_2)|}, \quad (13)$$

and $H(p, x)$ is defined as

$$H(p, x) = h(p, u = x, v) - h(p, u, v = x). \quad (14)$$

We use stochastic gradient decent (SGD) with learning rate adapted by AdaGrad [Duchi *et al.*, 2011] to update the word representations. The pseudo code for the proposed method is shown in Algorithm 1. Given a set of N relationally similar and dissimilar pattern-pairs, $\{(p_1^{(i)}, p_2^{(i)}, t(p_1^{(i)}, p_2^{(i)})\}_{i=1}^{N}$, Algorithm 1 initializes each word x_j in the vocabulary with a vector $\boldsymbol{x}_j \in \mathbb{R}^d$. The initialization can be conducted either using randomly sampled vectors from a zero mean and unit variance Gaussian distribution, or by pre-trained word representations. In our preliminary experiments, we found that the word vectors learnt by GloVe [Pennington *et al.*, 2014] to perform consistently well over random vectors when used as the initial word representations in the proposed method. Because word vectors trained using existing word representations already demonstrate a certain degree of relational structure with respect to proportional analogies, we believe that initializing using pre-trained word vectors assists the subsequent optimization process.

During each iteration, Algorithm 1 alternates between two steps. First, in Lines 3-5, it computes pattern representations using Eq. 4 from the current word representations for all the patterns (K in total) in the training dataset. Second, in Lines 6-10, for each train pattern-pair we compute the derivative of the loss according to Eq. 12, and update the word representations. These two steps are repeated for T iterations, after which the final set of word representations are returned.

The computational complexity of Algorithm 1 is $\mathcal{O}(TKd + TNMd)$, where d is the dimensionality of the word representations. Naively iterating over N training instances and M words in the vocabulary can be prohibitively expensive for large training datasets and vocabularies. However, in practice we can efficiently compute the updates using two tricks: *delayed updates* and *indexing*. Once we have computed the pattern representations for all K patterns in the first iteration, we can postpone the update of a representation for a pattern until that pattern next appears in a training instance. This reduces the number of patterns that are updated in each iteration to a maximum of 2 instead of K for the iterations $t > 1$. Because of the sparseness in co-occurrences, only a handful (ca. 100) of patterns co-occur with any given word-pair. Therefore, by pre-compiling an index from a pattern to the words with which that pattern co-occurs, we can limit the update of word representations in Line 8 to a much smaller number than M. Moreover, the vector subtraction can be parallized across the dimensions. Although the loss function defined by Eq. 5 is non-convex w.r.t. to word representations, in practice, Algorithm 1 converges after a few (less than 5) iterations. In practice, it requires less than an hour to train from a 2 billion word corpus where we have $N = 100,000$, $T = 10$, $K = 10,000$ and $M = 210,914$.

Lexical patterns contain sequences of multiple words. Therefore, exact occurrences of lexical patterns are rare compared to that of individual words even in large corpora. Directly learning representations for lexical patterns using their co-occurrence statistics leads to data sparseness issues, which becomes problematic when applying existing methods proposed for learning representations for single words to learn representations for lexical patterns that consist of multiple words. The proposal made in Eq. 4 to compute representations for patterns circumvent this data sparseness issue by indirectly modeling patterns through word representations.

3.1 Selecting Similar/Dissimilar Pattern-Pairs

We use the ukWaC corpus[1] to extract relationally similar (positive) and dissimilar (negative) pairs of patterns (p_i, p_j) to train the proposed method. The ukWaC is a 2 billion word corpus constructed from the Web limiting the crawl to the **.uk** domain. We select word-pairs that co-occur at least in 50 sentences within a co-occurrence window of 5 tokens. Moreover, using a stop word list, we ignore word-pairs that purely consists of stop words. We obtain $210,914$ word-pairs from this step. Next, we extract lexical patterns for those word-pairs by replacing the first and second word in a word-pair respectively by slot variables **X** and **Y** in a co-occurrence window of length 5 tokens to extract numerous lexical patterns. We select the top occurring $10,000$ lexical patterns (i.e. $K = 10,000$) for further processing.

We represent a pattern p by a vector where the elements correspond to the PPMI values $f(p, u, v)$ between p and all the word-pairs (u, v) that co-occur with p. Next, we compute the cosine similarity between all pairwise combinations of the $10,000$ patterns, and rank the pattern pairs in the descending order of their cosine similarities. We select the top ranked $50,000$ pattern-pairs as positive training instances. We select $50,000$ pattern-pairs from the bottom of the list which have non-zero similarity scores as negative training instances. The reason for not selecting pattern-pairs with zero similarity scores is that such patterns do not share any word-pairs in common, and are not informative as training data for updating word representations. Thus, the total number of training instances we select is $N = 50,000 + 50,000 = 100,000$.

4 Evaluating Word Representations using Proportional Analogies

To evaluate the ability of the proposed method to learn word representations that embed information related to semantic relations, we apply it to detect proportional analogies. For example, consider the proportional analogy, *man*:*woman* :: *king*:*queen*. Given, the first three words, a word representation learning method is required to find the fourth word from the vocabulary that maximizes the relational similarity between the two word-pairs in the analogy. Three benchmark datasets have been popularly used in prior work for evaluating analogies: **Google** dataset [Mikolov *et al.*, 2013c] ($10,675$ syntactic analogies and 8869 semantic analogies), **SemEval** dataset [Jurgens *et al.*, 2012] (79 questions), and **SAT** dataset [Turney, 2006] (374 questions). For the Google dataset, the set of candidates for the fourth word consists of all the words in the vocabulary. For the SemEval and SAT datasets, each question word-pair is assigned with a limited number of candidate word-pairs out of which only one is correct. The accuracy of a word representation is evaluated by the percentage of the correctly answered analogy questions out of all the questions in a dataset. We do not skip any questions in our evaluations.

Given a proportional analogy $a : b :: c : d$, we use the following measures proposed in prior work for measuring the relational similarity between (a, b) and (c, d).

[1] http://wacky.sslmit.unibo.it

Table 1: Word analogy results on benchmark datasets.

Method	sem.	synt.	total	SAT	SemEval
ivLBL CosAdd	63.60	61.80	62.60	20.85	34.63
ivLBL CosMult	65.20	63.00	64.00	19.78	33.42
ivLBL PairDiff	52.60	48.50	50.30	22.45	36.94
skip-gram CosAdd	31.89	67.67	51.43	29.67	40.89
skip-gram CosMult	33.98	69.62	53.45	28.87	38.54
skip-gram PairDiff	7.20	19.73	14.05	35.29	43.99
CBOW CosAdd	39.75	70.11	56.33	29.41	40.31
CBOW CosMult	38.97	70.39	56.13	28.34	38.19
CBOW PairDiff	5.76	13.43	9.95	33.16	42.89
GloVe CosAdd	86.67	82.81	84.56	27.00	40.11
GloVe CosMult	86.84	84.80	85.72	25.66	37.56
GloVe PairDiff	45.93	41.23	43.36	44.65	44.67
Prop CosAdd	86.70	85.35	85.97	29.41	41.86
Prop CosMult	**86.91**	**87.04**	**86.98**	28.87	39.67
Prop PairDiff	41.85	42.86	42.40	**45.99**	**44.88**

CosAdd proposed by Mikolov et al. [2013d] ranks candidates c according to the formula

$$\text{CosAdd}(a\!:\!b, c\!:\!d) = \cos(\boldsymbol{b} - \boldsymbol{a} + \boldsymbol{c}, \boldsymbol{d}), \qquad (15)$$

and selects the top-ranked candidate as the correct answer.

CosMult: CosAdd measure can be decomposed into the summation of three cosine similarities, where in practice one of the three terms often dominates the sum. To overcome this bias in CosAdd, Levy and Goldberg [2014] proposed the **CosMult** measure given by,

$$\text{CosMult}(a\!:\!b, c\!:\!d) = \frac{\cos(\boldsymbol{b}, \boldsymbol{d}) \cos(\boldsymbol{c}, \boldsymbol{d})}{\cos(\boldsymbol{a}, \boldsymbol{d}) + \epsilon}. \qquad (16)$$

We convert all cosine values $x \in [-1, 1]$ to positive values using the transformation $(x+1)/2$. Here, ϵ is a small constant value to prevent denominator becoming zero, and is set to 10^{-5} in the experiments.

PairDiff measures the cosine similarity between the two vectors that correspond to the difference of the word representations of the two words in each word-pair. It follows from our hypothesis that the semantic relation between two words can be represented by the vector difference of their word representations. PairDiff has been used by Mikolov et al. [2013d] for detecting semantic analogies and is given by,

$$\text{PairDiff}(a\!:\!b, c\!:\!d) = \cos(\boldsymbol{b} - \boldsymbol{a}, \boldsymbol{d} - \boldsymbol{c}). \qquad (17)$$

5 Experiments and Results

In Table 1, we compare the proposed method against previously proposed word representation learning methods: **ivLBL** [Mnih and Kavukcuoglu, 2013], **skip-gram** [Mikolov *et al.*, 2013c], **CBOW** [Mikolov *et al.*, 2013a], and **GloVe** [Pennington *et al.*, 2014]. All methods compared in Table 1 are trained on the same ukWaC corpus of 2B tokens to produce 300 dimensional word vectors. We use the publicly available implementations[2,3] by the original authors for

[2] https://code.google.com/p/word2vec/

[3] http://nlp.stanford.edu/projects/glove/

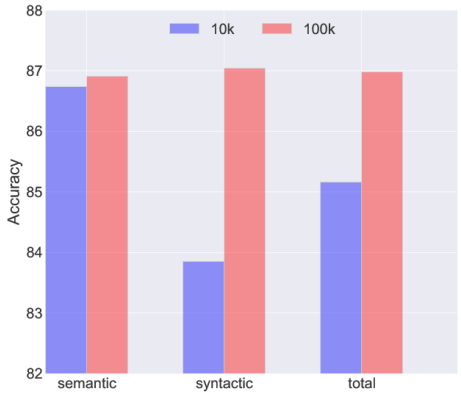

Figure 2: Accuracy on Google dataset when the proposed method is trained using 10k and 100k instances.

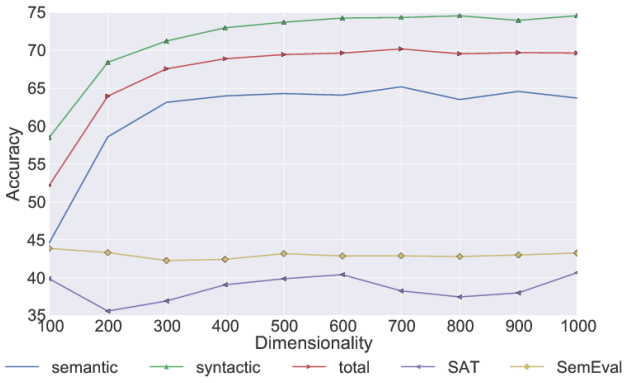

Figure 3: Accuracy of the proposed method on benchmark datasets for dimensionalities of the word representations.

training the word representations using the recommended parameter values. Therefore, any differences in performances reported in Table 1 can be directly attributable to the differences in the respective word representation learning methods. In all of our experiments, the proposed method converged with less than 5 iterations.

From Table 1 we see that the proposed method (denoted by **Prop**) achieves the best results for the semantic (**sem**), syntactic (**synt**) and their union (**total**) analogy questions in the Google dataset using CosMult measure. For analogy questions in SAT and SemEval datasets the best performance is reported by the proposed method using the PairDiff measure. The PairDiff measure computes the cosine similarity between the two difference vectors $\boldsymbol{b} - \boldsymbol{a}$ and $\boldsymbol{d} - \boldsymbol{c}$, ignoring the spatial distances between the individual words as opposed to CosAdd or CosMult. Recall that in the Google dataset we are required to find analogies from a large open vocabulary whereas in SAT and SemEval datasets the set of candidates is limited to a closed pre-defined set. Relying on direction alone, while ignoring spatial distance is problematic when considering the entire vocabulary as candidates because, we are likely to find candidates d that have the same relation to c as reflected by $\boldsymbol{a} - \boldsymbol{b}$. For example, given the analogy *man*:*woman*::*king*:*?*, we are likely to recover feminine entities, but not necessarily royal ones using PairDiff. On the other hand, in both SemEval and SAT datasets, the set of candidate answers already contains the related candidates, leaving mainly the direction to be decided. For the remainder of the experiments described in the paper, we use CosMult for evaluations on the Google dataset, whereas PairDiff is used for the SAT and SemEval datasets. Results reported in Table 1 reveal that according to the binomial exact test with 95% confidence the proposed method statistically significantly outperforms GloVe, the current state-of-the-art word representation learning method, on all three benchmark datasets.

To study the effect of the train dataset size on the performance of the proposed method, following the procedure described in Section 3.1, we sample two balanced datasets containing respectively $10,000$ and $100,000$ instances. Figure 2 shows the performance reported by the proposed method on the Google dataset. We see that the overall performance increases with the dataset size, and the gain is more for syntactic analogies. This result can be explained considering that semantic relations are more rare compared to syntactic relations in the ukWaC corpus, a generic web crawl, used in our experiments. However, the proposed train data selection method provides us with a potentially unlimited source of positive and negative training instances which we can use to further improve the performance.

To study the effect of the dimensionality d of the representation on the performance of the proposed method, we hold the train data size fixed and produce word representations for different dimensionalities. As shown in Figure 3, the performance increases until around 600 dimensions on the Google, and the SAT datasets after which it stabilizes. The performance on the SemEval dataset remains relatively unaffected by the dimensionality of the representation.

6 Conclusions

We proposed a method to learn word representations that embeds information related to semantic relations between words. A two step algorithm that alternates between pattern and word representations was proposed. The proposed method significantly outperforms the current state-of-the-art word representation learning methods on three datasets containing proportional analogies.

Semantic relations that can be encoded as attributes in words are only a fraction of all types of semantic relations. Whether we can accurately embed semantic relations that involve multiple entities, or semantic relations that are only extrinsically and implicitly represented remains unknown. We plan to explore these possibilities in our future work.

References

[Baroni and Lenci, 2010] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework

for corpus-based semantics. *Computational Linguistics*, 36(4):673 – 721, 2010.

[Baroni *et al.*, 2014] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL'14*, pages 238–247, 2014.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137 – 1155, 2003.

[Bollegala *et al.*, 2007a] D. Bollegala, Y. Matsuo, and M. Ishizuka. An integrated approach to measuring semantic similarity between words using information available on the web. In *Proceedings of NAACL HLT*, pages 340–347, 2007.

[Bollegala *et al.*, 2007b] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Websim: A web-based semantic similarity measure. In *Proc. of 21st Annual Conference of the Japanese Society of Artitificial Intelligence*, pages 757 – 766, 2007.

[Bollegala *et al.*, 2010] Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *WWW 2010*, pages 151 – 160, 2010.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNP'14*, pages 1724–1734, 2014.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuska. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493 – 2537, 2011.

[Duc *et al.*, 2010] Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. Using relational similarity between word pairs for latent relational search on the web. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 196 – 199, 2010.

[Duc *et al.*, 2011] Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka. Cross-language latent relational search: Mapping knowledge across languages. In *Proc. of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1237 – 1242, 2011.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121 – 2159, July 2011.

[Hashimoto *et al.*, 2014] Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. Jointly learning word representations and composition functions using predicate-argument structures. In *EMNLP'14*, pages 1544–1555, 2014.

[Huang *et al.*, 2012] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Improving word representations via global context and multiple word prototypes. In *ACL'12*, pages 873 – 882, 2012.

[Jurgens *et al.*, 2012] David A. Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. Measuring degrees of relational similarity. In *SemEval'12*, 2012.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, 2014.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, and Jeffrey Dean. Efficient estimation of word representation in vector space. *CoRR*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv*, 2013.

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

[Mikolov *et al.*, 2013d] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continous space word representations. In *NAACL'13*, pages 746 – 751, 2013.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, 2013.

[Pennington *et al.*, 2014] Jeffery Pennington, Richard Socher, and Christopher D. Manning. Glove: global vectors for word representation. In *EMNLP*, 2014.

[Socher *et al.*, 2011] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP'11*, pages 151–161, 2011.

[Turian *et al.*, 2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384 – 394, 2010.

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

[Turney, 2005] P.D. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141, 2005.

[Turney, 2006] P.D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.