Think Globally, Embed Locally — Locally Linear Meta-embedding of Words

Danushka Bollegala[*§] **Kohei Hayashi**[†§] **Ken-ichi Kawarabayashi**[‡§]
The University of Liverpool, Liverpool, United Kingdom.[*]
Japan Advanced Institute of Science and Technology, Tokyo, Japan.[†]
National Institute of Informatics, Tokyo, Japan. [‡]
Japan Science and Technology Agency, ERATO, Kawarabayashi Large Graph Project.[§]

Abstract

Distributed word embeddings have shown superior performances in numerous Natural Language Processing (NLP) tasks. However, their performances vary significantly across different tasks, implying that the word embeddings learnt by those methods capture complementary aspects of lexical semantics. Therefore, we believe that it is important to combine the existing word embeddings to produce more accurate and complete *meta-embeddings* of words. For this purpose, we propose an unsupervised locally linear meta-embedding learning method that takes pre-trained word embeddings as the input, and produces more accurate meta embeddings. Unlike previously proposed meta-embedding learning methods that learn a global projection over all words in a vocabulary, our proposed method is sensitive to the differences in local neighbourhoods of the individual source word embeddings. Moreover, we show that vector concatenation, a previously proposed highly competitive baseline approach for integrating word embeddings, can be derived as a special case of the proposed method. Experimental results on semantic similarity, word analogy, relation classification, and short-text classification tasks show that our meta-embeddings to significantly outperform prior methods in several benchmark datasets, establishing a new state of the art for meta-embeddings.

1 Introduction

Representing the meanings of words is a fundamental task in Natural Language Processing (NLP). One popular approach to represent the meaning of a word is to *embed* it in some fixed-dimensional vector space (Turney and Pantel, 2010). In contrast to sparse and high-dimensional counting-based distributional word representation methods that use co-occurring contexts of a word as its representation (Baroni, Dinu, and Kruszewski, 2014), dense and low-dimensional prediction-based distributed word representations have obtained impressive performances in numerous NLP tasks such as sentiment classification (Socher et al., 2013), and machine translation (Zou et al., 2013). Several distributed word embedding learning methods based on different learning strategies have been proposed (Pennington, Socher, and Manning, 2014; Mikolov, Chen, and Dean, 2013; Huang et al., 2012; Collobert and Weston, 2008; Mnih and Hinton, 2009).

Previous works studying the differences in word embedding learning methods (Chen et al., 2013; Yin and Schütze, 2016) have shown that word embeddings learnt using different methods and from different resources have significant variation in quality and characteristics of the semantics captured. For example, Hill et al. (2014, 2015) showed that the word embeddings trained from monolingual vs. bilingual corpora capture different local neighbourhoods. Bansal, Gimpel, and Livescu (2014) showed that an ensemble of different word representations improves the accuracy of dependency parsing, implying the complementarity of the different word embeddings. This suggests the importance of *meta-embedding* – creating a new embedding by combining different existing embeddings. We refer to the input word embeddings to the meta-embedding process as the *source embeddings*. Yin and Schütze (2016) showed that by meta-embedding five different pre-trained word embeddings, we can overcome the out-of-vocabulary problem, and improve the accuracy of cross-domain part-of-speech (POS) tagging. Encouraged by the above-mentioned prior results, we expect an ensemble containing multiple word embeddings to produce better performances than the constituent individual embeddings in NLP tasks.

There are three main challenges a meta-embedding learning method must overcome.

First, the vocabularies covered by the source embeddings might be different because they have been trained on different text corpora. Therefore, not all words will be equally represented by all the source embeddings. Even in situations where the implementations of the word embedding learning methods are publicly available, it might not be possible to retrain those embeddings because the text corpora on which those methods were originally trained might not be publicly available. Moreover, it is desirable if the meta-embedding method does not require the original resources upon which they were trained such as corpora or lexicons, and can directly work with the pre-trained word embeddings. This is particularly attractive from a computational point of view because re-training source embedding methods on large corpora might require significant processing times and resources.

Second, the vector spaces and their dimensionalities of the source embeddings might be different. In most prediction-based word embedding learning methods the word vec-

tors are randomly initialised. Therefore, there is no obvious correspondence between the dimensions in two word embeddings learnt even from two different runs of the same method, let alone from different methods (Tian et al., 2016). Moreover, the pre-trained word embeddings might have different dimensionalities, which is often a hyperparameter set experimentally. This becomes a challenging task when incorporating multiple source embeddings to learn a single meta-embedding because the alignment between the dimensionalities of the source embeddings is unknown.

Third, the local neighbourhoods of a particular word under different word embeddings show a significant diversity. For example, as the nearest neighbours of the word *bank*, GloVe (Pennington, Socher, and Manning, 2014), a word sense insensitive embedding, lists *credit, financial, cash*, whereas word sense sensitive embeddings created by Huang et al. (2012) lists *river, valley, marsh* when trained on the same corpus. We see that the nearest neighbours for the different senses of the word bank (i.e. financial institution vs. river bank) are captured by the different word embeddings. Meta-embedding learning methods that learn a single global projection over the entire vocabulary are insensitive to such local variations in the neighbourhoods (Yin and Schütze, 2016).

To overcome the above-mentioned challenges, we propose a *locally-linear* meta-embedding learning method that (a) requires only the words in the vocabulary of each source embedding, without having to predict embeddings for missing words, (b) can meta-embed source embeddings with different dimensionalities, (c) is sensitive to the diversity of the neighbourhoods of the source embeddings.

Our proposed method comprises of two steps: a neighbourhood *reconstruction step* (Section 3.2), and a *projection step* (Section 3.3). In the reconstruction step, we represent the embedding of a word by the linearly weighted combination of the embeddings of its nearest neighbours in each source embedding space. Although the number of words in the vocabulary of a particular source embedding can be potentially large, the consideration of nearest neighbours enables us to limit the representation to a handful of parameters per each word, not exceeding the neighbourhood size. The weights we learn are shared across different source embeddings, thereby incorporating the information from different source embeddings in the meta-embedding. Interestingly, vector concatenation, which has found to be an accurate meta-embedding method, can be derived as a special case of this reconstruction step.

Next, the projection step computes the meta-embedding of each word such that the nearest neighbours in the source embedding spaces are embedded closely to each other in the meta-embedding space. The reconstruction weights can be efficiently computed using stochastic gradient descent, whereas the projection can be efficiently computed using a truncated eigensolver.

It is noteworthy that we do not directly compare different source embeddings for the same word in the reconstruction step nor in the projection step. This is important because the dimensions in source word embeddings learnt using different word embedding learning methods are not aligned. Moreover, a particular word might not be represented by all source embeddings. This property of the proposed method is attractive because it obviates the need to align source embeddings, or predict missing source word embeddings prior to meta-embedding. Therefore, all three challenges described above are solved by the proposed method.

The above-mentioned properties of the proposed method enables us to compute meta-embeddings for five different source embeddings covering 2.7 million unique words. We evaluate the meta-embeddings learnt by the proposed method on semantic similarity prediction, analogy detection, relation classification, and short-text classification tasks. The proposed method significantly outperforms several competitive baselines and previously proposed meta-embedding learning methods (Yin and Schütze, 2016) on multiple benchmark datasets.

2 Related Work

Yin and Schütze (2016) proposed a meta-embedding learning method (1TON) that projects a meta-embedding of a word into the source embeddings using separate projection matrices. The projection matrices are learnt by minimising the sum of squared Euclidean distance between the projected source embeddings and the corresponding original source embeddings for all the words in the vocabulary. They propose an extension (1TON+) to their meta-embedding learning method that first predicts the source word embeddings for out-of-vocabulary words in a particular source embedding, using the known word embeddings. Next, 1TON method is applied to learn the meta-embeddings for the union of the vocabularies covered by all of the source embeddings.

Experimental results in semantic similarity prediction, word analogy detection, and cross-domain POS tagging tasks show the effectiveness of both 1TON and 1TON+. In contrast to our proposed method which learns locally-linear projections that are sensitive to the variations in the local neighbourhoods in the source embeddings, 1TON and 1TON+ can be seen as globally linear projections between meta and source embedding spaces. As we see later in Section 4.4, our proposed method outperforms both of those methods consistently in all benchmark tasks demonstrating the importance of neighbourhood information when learning meta-embeddings. Moreover, our proposed meta-embedding method does not directly compare different source embeddings, thereby obviating the need to predict source embeddings for out-of-vocabulary words. Locally-linear embeddings are attractive from a computational point-of-view as well because during optimisation we require information from only the local neighbourhood of each word.

Although not learning any meta-embeddings, several prior work have shown that by incorporating multiple word embeddings learnt using different methods improve performance in various NLP tasks. For example, Tsuboi (2014) showed that by using both word2vec and GloVe embeddings together in a POS tagging task, it is possible to improve the tagging accuracy, if we had used only one of those embeddings. Similarly, Turian, Ratinov, and Bengio (2010) collectively used Brown clusters, CW and HLBL embeddings, to

improve the performance of named entity recognition and chucking tasks.

Luo et al. (2014) proposed a multi-view word embedding learning method that uses a two-sided neural network. They adapt pre-trained CBOW (Mikolov et al., 2013) embeddings from Wikipedia and click-through data from a search engine. Their problem setting is different from ours because their source embeddings are trained using the same word embedding learning method but on different resources whereas, we consider source embeddings trained using different word embedding learning methods and resources. Although their method could be potentially extended to meta-embed different source embeddings, the unavailability of their implementation prevented us from exploring this possibility.

Goikoetxea, Agirre, and Soroa (2016) showed that concatenation of word embeddings learnt separately from a corpus and the WordNet to produce superior word embeddings. Moreover, performing Principal Component Analysis (PCA) on the concatenated embeddings slightly improved the performance on word similarity tasks. In Section 4.3, we discuss the relationship between the proposed method and vector concatenation.

3 Locally Linear Meta-Embeddings

3.1 Problem Settings

To explain the proposed meta-embedding learning method, let us consider two source word embeddings, denoted by $\mathcal{S}_1$ and $\mathcal{S}_2$. Although we limit our discussion here to two source embeddings for the simplicity of the description, the proposed meta-embedding learning method can be applied to any number of source embeddings. Indeed in our experiments we consider five different source embeddings. Moreover, the proposed method is not limited to meta-embedding unigrams, and can be used for n-grams of any length n, provided that we have source embeddings for those n-grams.

We denote the dimensionalities of $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively by d_1 and d_2 (in general, $d_1 \neq d_2$). The sets of words covered by each source embedding (i.e. vocabulary) are denoted by $\mathcal{V}_1$ and $\mathcal{V}_2$. The source embedding of a word $v \in \mathcal{V}_1$ is represented by a vector $\boldsymbol{v}^{(1)} \in \mathbb{R}^{d_1}$, whereas the same for a word $v \in \mathcal{V}_2$ by a vector $\boldsymbol{v}^{(2)} \in \mathbb{R}^{d_2}$. Let the set union of $\mathcal{V}_1$ and $\mathcal{V}_2$ be $\mathcal{V} = \{v_1, \ldots, v_n\}$ containing n words. In particular, note that our proposed method does not require a word v to be represented by all source embeddings, and can operate on the union of the vocabularies of the source embeddings. The meta-embedding learning problem is then to learn an embedding $\boldsymbol{v}^{(\mathcal{P})} \in \mathbb{R}^{d_\mathcal{P}}$ in a meta-embedding space $\mathcal{P}$ with dimensionality $d_\mathcal{P}$ for each word $v \in \mathcal{V}$.

For a word v, we denote its k-nearest neighbour set in embedding spaces $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively by $\mathcal{N}_1(v)$ and $\mathcal{N}_2(v)$ (in general, $|\mathcal{N}_1(v)| \neq |\mathcal{N}_2(v)|$). As discussed already in Section 1, different word embedding methods encode different aspects of lexical semantics, and are likely to have different local neighbourhoods. Therefore, by requiring the meta embedding to consider different neighbourhood constraints in the source embedding spaces we hope to exploit the complementarity in the source embeddings.

3.2 Nearest Neighbour Reconstruction

The first-step in learning a locally linear meta-embedding is to reconstruct each source word embedding using a linearly weighted combination of its k-nearest neighbours. Specifically, we construct each word $v \in \mathcal{V}$ separately from its k-nearest neighbours $\mathcal{N}_1(v)$, and $\mathcal{N}_2(v)$. The reconstruction weight w_{vu} assigned to a neighbour $u \in \mathcal{N}_1(v) \cup \mathcal{N}_2(v)$ is found by minimising the reconstruction error $\Phi(\mathbf{W})$ defined by (1), which is the sum of local distortions in the two source embedding spaces.

$$\Phi(\mathbf{W}) = \sum_{i=1}^{2} \sum_{v \in \mathcal{V}} \left\| \boldsymbol{v}^{(i)} - \sum_{u \in \mathcal{N}_i(v)} w_{vu} \boldsymbol{u}^{(i)} \right\|_2^2 \quad (1)$$

Words that are not k-nearest neighbours of v in either of the source embedding spaces will have their weights set to zero (i.e. $w_{vu} = 0, \forall u \notin \mathcal{N}_1(v) \cup \mathcal{N}_2(v)$). Moreover, we require the sum of reconstruction weights for each v to be equal to one (i.e. $\sum_{u \in \mathcal{V}} w_{uv} = 1$).

To compute the weights w_{vu} that minimise (1), we compute its error gradient $\frac{\partial \Phi(\mathbf{W})}{\partial w_{vu}}$ as follows:

$$-2 \sum_{i=1}^{2} \left(\boldsymbol{v}^{(i)} - \sum_{x \in \mathcal{N}_i(v)} w_{vx} \boldsymbol{x}^{(i)} \right)^\top \boldsymbol{u}^{(i)} \mathbb{I}[u \in \mathcal{N}_i(v)]$$

Here, the indicator function, $\mathbb{I}[x]$, returns 1 if x is true and 0 otherwise. We uniformly randomly initialise the weights w_{vu} for each neighbour u of v, and use stochastic gradient descent (SGD) with the learning rate scheduled by AdaGrad (Duchi, Hazan, and Singer, 2011) to compute the optimal values of the weights. The initial learning rate is set to 0.01 and the maximum number of iterations to 100 in our experiments. Empirically we found that these settings to be adequate for convergence. Finally, we normalise the weights w_{uv} for each v such that they sum to 1 (i.e. $\sum_{u \in \mathcal{V}} w_{vu} = 1$).

Exact computation of k nearest neighbours for a given data point in a set of n points requires all pairwise similarity computations. Because we must repeat this process for each data point in the set, this operation would require a time complexity of $\mathcal{O}(n^3)$. This is prohibitively large for the vocabularies we consider in NLP where typically $n > 10^3$. Therefore, we resort to approximate methods for computing k nearest neighbours. Specifically, we use the BallTree algorithm (Kibriya and Frank, 2007) to efficiently compute the approximate k-nearest neighbours, for which the time complexity of tree construction is $\mathcal{O}(n \log n)$ for n data points.

The solution to the least square problem given by (1) subjected to the summation constraints can be found by solving a set of linear equations. Time complexity of this step is $\mathcal{O}(N(d_1|\mathcal{N}_1|^3 + d_2|\mathcal{N}_2|^3))$, which is cubic in the neighbourhood size and linear in both the dimensionalities of the embeddings and vocabulary size. However, we found that the iterative estimation process using SGD described above to be more efficient in practice. Because k is significantly smaller than the number of words in the vocabulary, and often the word being reconstructed is contained in the neighbourhood, the reconstruction weight computation converges

after a small number (less than 5 in our experiments) of iterations.

3.3 Projection to Meta-Embedding Space

In the second step of the proposed method, we compute the meta-embeddings $\boldsymbol{v}^{(\mathcal{P})}, \boldsymbol{u}^{(\mathcal{P})} \in \mathbb{R}^{d_{\mathcal{P}}}$ for words $v, u \in \mathcal{V}$ using the reconstruction weights w_{vu} we computed in Section 3.2. Specifically, the meta-embeddings must minimise the projection cost, $\Psi(\mathcal{P})$, defined by (2).

$$\Psi(\mathcal{P}) = \sum_{v \in \mathcal{V}} \left\| \boldsymbol{v}^{(\mathcal{P})} - \sum_{i=1}^{2} \sum_{u \in \mathcal{N}_i(v)} w_{vu} \boldsymbol{u}^{(\mathcal{P})} \right\|_2^2 \quad (2)$$

By finding a $\mathcal{P}$ space that minimises (2), we hope to preserve the rich neighbourhood diversity in all source embeddings within the meta-embedding. The two summations in (2) over $N_1(v)$ and $N_2(v)$ can be combined to re-write (2) as follows:

$$\Psi(\mathcal{P}) = \sum_{v \in \mathcal{V}} \left\| \boldsymbol{v}^{(\mathcal{P})} - \sum_{u \in \mathcal{N}_1(v) \cup \mathcal{N}_2(v)} w'_{vu} \boldsymbol{u}^{(\mathcal{P})} \right\|_2^2 \quad (3)$$

Here, w'_{uv} is computed using (4).

$$w'_{vu} = w_{vu} \sum_{i=1}^{2} \mathbb{I}[u \in \mathcal{N}_i(v)] \quad (4)$$

The $d_{\mathcal{P}}$ dimensional meta-embeddings are given by the eigenvectors corresponding to the smallest $(d_{\mathcal{P}} + 1)$ eigenvectors of the matrix $\mathbf{M}$ given by (5).

$$\mathbf{M} = (\mathbf{I} - \mathbf{W}')^{\top} (\mathbf{I} - \mathbf{W}') \quad (5)$$

Here, $\mathbf{W}'$ is a matrix with the (v, u) element set to w'_{vu}. The smallest eigenvalue of $\mathbf{M}$ is zero and the corresponding eigenvector is discarded from the projection. The eigenvectors corresponding to the next smallest $d_{\mathcal{P}}$ eigenvalues of the symmetric matrix $\mathbf{M}$ can be found without performing a full matrix diagonalisation (Bai, 2000). Operations involving $\mathbf{M}$ such as the left multiplication by $\mathbf{M}$, which is required by most sparse eigensolvers, can exploit the fact that $\mathbf{M}$ is expressed in (5) as the product between two sparse matrices. Moreover, truncated randomised methods (Halko, Martinsson, and Tropp, 2010) can be used to find the smallest eigenvectors, without performing full eigen decompositions. In our experiments, we set the neighbourhood sizes for all words in all source embeddings equal to n (i.e $\forall i \; |\mathcal{N}_i(v)| = N, \forall v \in \mathcal{V}$), and project to a $d_{\mathcal{P}} (< N)$ dimensional meta-embedding space.

4 Experiments and Results

4.1 Source Word Embeddings

We use five previously proposed pre-trained word embedding sets as the source embeddings in our experiments:

(a) **HLBL** – hierarchical log-bilinear (Mnih and Hinton, 2009) embeddings released by Turian, Ratinov, and Bengio (2010) (246,122 word embeddings, 100 dimensions, trained on Reuters Newswire (RCV1) corpus),

(b) **Huang** – Huang et al. (2012) used global contexts to train multi-prototype word embeddings that are sensitive to word senses (100,232 word embeddings, 50 dimensions, trained on April 2010 snapshot of Wikipedia),

(c) **GloVe** – Pennington, Socher, and Manning (2014) used global co-occurrences of words over a corpus to learn word embeddings (1,193,514 word embeddings, 300 dimensions, trained on 42 billion corpus of web crawled texts),

(d) **CW** – Collobert and Weston (2008) learnt word embeddings following a multitask learning approach covering multiple NLP tasks (we used the version released by (Turian, Ratinov, and Bengio, 2010) trained on the same corpus as **HLBL** containing 268,810 word embeddings, 200 dimensions),

(e) **CBOW** – Mikolov et al. (2013) proposed the continuous bag-of-words method to train word embeddings (we discarded phrase embeddings and selected 929,922 word embeddings, 300 dimensions, trained on the Google News corpus containing ca. 100 billion words).

The intersection of the five vocabularies is 35,965 words, whereas their union is 2,788,636. Although any word embedding can be used as a source we select the above-mentioned word embeddings because (a) our goal in this paper is *not* to compare the differences in performance of the source embeddings, and (b) by using the same source embeddings as in prior work (Yin and Schütze, 2016), we can perform a fair evaluation.[1] In particular, we could use word embeddings trained by the same algorithm but on different resources, or different algorithms on the same resources as the source embeddings. We defer such evaluations to an extended version of this conference submission.

4.2 Evaluation Tasks

The standard protocol for evaluating word embeddings is to use the embeddings in some NLP task and to measure the relative increase (or decrease) in performance in that task. We use four such extrinsic evaluation tasks:

Semantic similarity measurement: We measure the similarity between two words as the cosine similarity between the corresponding embeddings, and measure the Spearman correlation coefficient against the human similarity ratings. We use Rubenstein and Goodenough's dataset (Rubenstein and Goodenough, 1965) (**RG**, 65 word-pairs), rare words dataset (**RW**, 2034 word-pairs) (Luong, Socher, and Manning, 2013), Stanford's contextual word similarities (**SCWS**, 2023 word-pairs) (Huang et al., 2012), the **MEN** dataset (3000 word-pairs) (Bruni et al., 2012), and the SimLex dataset (Hill, Reichart, and Korhonen, 2015) (**SL** 999 word-pairs).

In addition, we use the Miller and Charles' dataset (Miller and Charles, 1998) (**MC**, 30 word-pairs) as a validation dataset to tune various hyperparameters such as the

[1] Although skip-gram embeddings are shown to outperform most other embeddings, they were not used as a source by Yin and Schütze (2016). Therefore, to be consistent in comparisons against prior work, we decided not to include skip-gram as a source.

neighbourhood size, and the dimensionality of the meta-embeddings for the proposed method and baselines.

Word analogy detection: Using the CosAdd method, we solve word-analogy questions in the Google dataset (**GL**) (Mikolov et al., 2013) (19544 questions), and in the SemEval (**SE**) dataset (Jurgens et al., 2012). Specifically, for three given words a, b and c, we find a fourth word d that correctly answers the question *a to b is c to what?* such that the cosine similarity between the two vectors $(\boldsymbol{b} - \boldsymbol{a} + \boldsymbol{c})$ and $\boldsymbol{d}$ is maximised.

Relation classification: We use the DiffVec (**DV**) (Vylomova et al., 2016) dataset containing 12,458 triples of the form $(\text{relation}, \text{word}_1, \text{word}_2)$ covering 15 relation types. We train a 1-nearest neighbour classifer where for each target tuple we measure the cosine similarity between the vector offset for its two word embeddings, and those of the remaining tuples in the dataset. If the top ranked tuple has the same relation as the target tuple, then it is considered to be a correct match. We compute the (micro-averaged) classification accuracy over the entire dataset as the evaluation measure.

Short-text classification: We use two binary short-text classification datasets: Stanford sentiment treebank (**TR**)[2] (903 positive test instances and 903 negative test instances), and the movie reviews dataset (**MR**) (Pang and Lee, 2005) (5331 positive instances and 5331 negative instances). Each review is represented as a bag-of-words and we compute the centroid of the embeddings of the words in each bag to represent that review. Next, we train a binary logistic regression classifier with a cross-validated ℓ_2 regulariser using the train portion of each dataset, and evaluate the classification accuracy using the test portion of the dataset.

4.3 Baselines

Concatenation (CONC): A simple baseline method for combining pre-trained word embeddings is to concatenate the embedding vectors for a word w to produce a meta-embedding for w. Each source embedding of w is ℓ_2 normalised prior to concatenation such that each source embedding contributes equally (a value in $[-1, 1]$) when measuring the word similarity using the dot product. As also observed by Yin and Schütze (2016) we found that **CONC** performs poorly without emphasising **GloVe** and **CBOW** by a constant factor (which is set to 8 using **MC** as a validation dataset) when used in conjunction with **HLBL**, **Huang**, and **CW** source embeddings.

Interestingly, concatenation can be seen as a special case in the reconstruction step described in Section 3.2. To see this, let us denote the concatenation of column vectors $\boldsymbol{v}^{(1)}$ and $\boldsymbol{v}^{(2)}$ by $\boldsymbol{x} = (\boldsymbol{v}^{(1)}; \boldsymbol{v}^{(2)})$, and $\boldsymbol{u}^{(1)}$ and $\boldsymbol{u}^{(2)}$ by $\boldsymbol{y} = (\boldsymbol{u}^{(1)}; \boldsymbol{u}^{(2)})$, where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{d_1+d_2}$. Then, the reconstruction error defined by (1) can be written as follows:

$$\Phi(\mathbf{W}) = \sum_{v \in \mathcal{V}} \left|\left| \boldsymbol{x} - \sum_{u \in \mathcal{N}(v)} w_{vu} \boldsymbol{y} \right|\right|_2^2 \quad (6)$$

[2]http://nlp.stanford.edu/sentiment/treebank.html

Here, the vocabulary $\mathcal{V}$ is constrained to the intersection $\mathcal{V}_\mathcal{S} \cap \mathcal{V}_\mathcal{T}$ because concatenation is not defined for missing words in a source embedding. Alternatively, one could use zero-vectors for missing words or (better) predict the word embeddings for missing words prior to concatenation. However, we consider such extensions to be beyond the simple concatenation baseline we consider here.[3] On the other hand, the common neighbourhood $\mathcal{N}(v)$ in (6) can be obtained by either limiting $\mathcal{N}(v)$ to $\mathcal{N}_1(v) \cap \mathcal{N}_2(v)$ or, by extending the neighbourhoods to the entire vocabulary ($\mathcal{N}(v) = \mathcal{V}$). (6) shows that under those neighbourhood constraints, the first step in our proposed method can be seen as reconstructing the neighbourhood of the concatenated space. The second step would then find meta-embeddings that preserve the locally linear structure in the concatenated space.

One drawback of concatenation is that it increases the dimensionality of the meta-embeddings compared to the source-embeddings, which might be problematic when storing or processing the meta-embeddings (for example, for the five source embeddings we use here $d_\mathcal{P} = 100 + 50 + 300 + 200 + 300 = 950$).

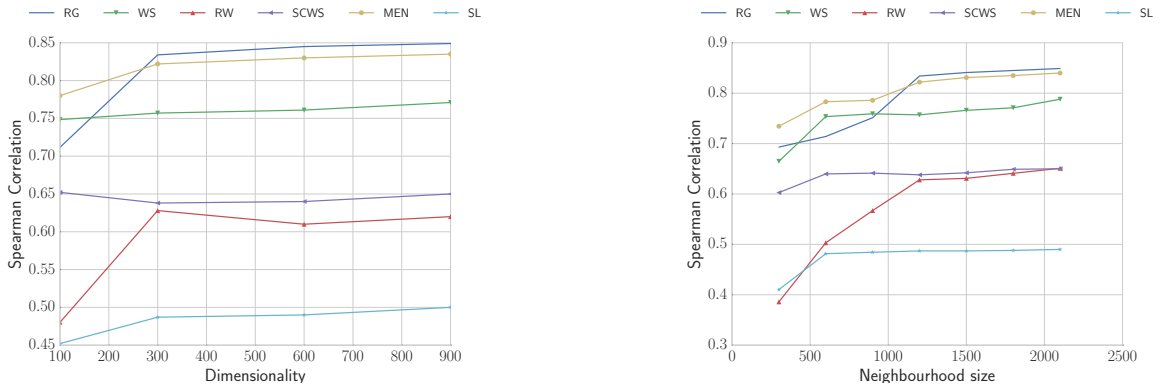
Singular Value Decomposition (SVD): We create an $N \times 950$ matrix $\mathbf{C}$ by arranging the **CONC** vectors for the union of all source embedding vocabularies. For words that are missing in a particular source embedding, we assign zero vectors of that source embedding's dimensionality. Next, we perform SVD on $\mathbf{C} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and the diagonal matrix $\mathbf{D}$ contains the singular values of $\mathbf{C}$. We then select the d largest left singular vectors from $\mathbf{U}$ to create a $d_\mathcal{P}$ dimensional embeddings for the N words. Using the **MC** validation dataset, we set $d_\mathcal{P} = 300$. Multiplying $\mathbf{U}$ by the singular values, a technique used to weight the latent dimensions considering the salience of the singular values, did not result in any notable improvements in our experiments.

4.4 Meta-Embedding Results

Using the **MC** dataset, we find the best values for the neighbourhood size $n = 1200$ and dimensionality $d_\mathcal{P} = 300$ for the **Proposed** method. We plan to publicly release our meta-embeddings on acceptance of the paper.

We summarise the experimental results for different methods on different tasks/datasets in Table 1. In Table 1, rows 1-5 show the performance of the individual source embeddings. Next, we perform ablation tests (rows 6-20) where we hold-out one source embedding and use the other four with each meta-embedding method. We evaluate statistical significance against best performing individual source embedding on each dataset. For the semantic similarity benchmarks we use Fisher transformation to compute $p < 0.05$ confidence intervals for Spearman correlation coefficients. In all other (classification) datasets, we used Clopper-Pearson binomial exact confidence intervals at $p < 0.05$.

[3]Missing words does *not* affect the performance of **CONC** because all words in the benchmark datasets we use in our experiments are covered by all source embeddings.



(a) (b)

Figure 1: Performance vs. dimensionality (neighbourhood size fixed at 1200) shown in (a), and vs. neighbourhood size (dimensionality fixed at 300) shown in (b) for meta embedding learning.

		Model	RG	MC	WS	RW	SCWS	MEN	SL	GL	SE	DV	SA	MR
sources	1	GloVe	81.7	80.8	64.3	38.4	54.0	74.3	37.4	70.5	39.9	87.7	73.4	70.0
	2	CBOW	76.0	82.2	69.8	53.4	53.4	78.2	44.2	75.2	39.1	87.4	73.6	71.0
	3	HLBL	35.3	49.3	35.7	19.1	47.7	30.7	22.1	16.6	34.8	72.0	62.6	61.6
	4	Huang	51.3	58.8	58.0	36.4	63.7	56.1	21.7	8.3	35.2	76.0	64.8	60.9
	5	CW	29.9	34.3	28.4	15.3	39.8	25.7	15.6	4.7	34.6	75.6	62.7	61.4
ablation	6	CONC (-GloVe)	75.0	79.0	70.0	55.3	62.9	77.7	41.5	64.0	38.7	82.9	72.1	69.1
	7	CONC (-CBOW)	80.8	81.0	65.2	46.0	56.3	74.9	37.3	70.0	38.8	86.0	71.6	69.9
	8	CONC (-HLBL)	83.0	84.0	71.9	53.4	61.4	80.1*	41.6	72.7	39.5	84.9	71.0	69.4
	9	CONC (-Huang)	83.0	84.0	71.6	48.8	60.8	80.1*	41.9	72.8	40.0	86.7	71.2	69.1
	10	CONC (-CW)	82.9	84.0	71.9	53.3	61.6	80.2*	41.6	72.6	39.6	84.9	72.3	69.9
	11	SVD (-GloVe)	78.6	79.9	68.4	53.9	61.6	77.5	40.1	61.7	38.5	84.1	71.6	69.8
	12	SVD (-CBOW)	80.5	81.2	64.4	45.3	55.3	74.2	35.7	70.9	38.7	86.7	73.4	69.1
	13	SVD (-HLBL)	82.7	83.6	70.3	52.6	60.1	79.9*	39.6	73.5	39.8	87.3	73.2	70.4
	14	SVD (-Huang)	82.5	85.0	70.3	48.6	59.8	79.9*	39.9	73.7	40.0	87.3	73.5	70.8
	15	SVD (-CW)	82.5	83.9	70.4	52.5	60.1	80.0*	39.7	73.3	39.8	87.2	73.1	70.7
	16	Proposed (-GloVe)	79.8	79.7	71.1	54.7	62.3	78.2	46.1	84.2*	39.8	85.4	72.2	70.2
	17	Proposed (-CBOW)	80.9	82.1	67.4	58.7*	58.7	75.7	45.2	85.2*	40.1	87.1	73.8	70.1
	18	Proposed (-HLBL)	82.1	86.1	71.3	58.3*	62.1	81.9*	34.8	86.3*	40.3	87.7	73.7	71.1
	19	Proposed (-Huang)	81.2	85.2	73.1	55.1	63.7	81.4*	42.3	82.6*	41.1	87.5	73.9	71.2
	20	Proposed (-CW)	83.1	84.8	72.5	58.5	62.3	81.1*	43.5	88.4*	41.9	87.8	71.6	71.1
ensemble	21	CONC	82.9	84.1	71.9	53.3	61.5	80.2*	41.6	72.9	39.6	84.9	72.4	69.9
	22	SVD	82.7	83.9	70.4	52.6	60.0	79.9*	39.7	73.4	39.7	87.2	73.4	70.7
	23	1TON	80.7	80.7	74.5	60.1*	61.6	73.5	46.4	76.8	42.3*	87.6	73.8	70.3
	24	1TON+	82.7	85.0	75.3	61.6*	60.2	74.1	46.3	77.0	40.1	83.9	73.9	69.2
	25	Proposed	**83.4**	**86.2**	**75.7***	**62.8***	**63.8**	**82.2***	**48.7**	**89.9***	**43.1***	**88.7***	**74.0**	**71.3**

Table 1: Results on word similarity, analogy, relation and short-text classification tasks. For each task, the best performing method is shown in bold. Statistically significant improvements over the best individual source embedding are indicated by an asterisk.

Among the individual source embeddings, we see that **GloVe** and **CBOW** stand out as the two best embeddings. This observation is further confirmed from ablation results, where the removal of **GloVe** or **CBOW** often results in a decrease in performance. Performing SVD (rows 11-15) after concatenating, does not always result in an improvement. SVD is a global projection that reduces the dimensionality of the meta-embeddings created via concatenation. This result indicates that different source embeddings might require different levels of dimensionality reductions, and applying a single global projection does not always guarantee improvements. Ensemble methods that use all five source embeddings are shown in rows 21-25. **1TON** and **1TON+** are proposed by Yin and Schütze (2016), and were detailed in Section 2. Because they did not evaluate on all tasks that we do here, to conduct a fair and consistent evaluation we used their publicly available meta-embeddings[4] without retraining by ourselves.

Overall, from Table 1, we see that the **Proposed** method (row 25) obtains the best performance in *all* tasks/datasets. In 6 out of 12 benchmarks, this improvement is statistically significant over the best single source embedding. Moreover,

[4]http://cistern.cis.lmu.de/meta-emb/

in the **MEN** dataset (the largest among the semantic similarity benchmarks compared in Table 1 with 3000 word-pairs), and the **Google** dataset, the improvements of the **Proposed** method over the previously proposed **1TON** and **1TON+** are statistically significant.

The ablation results for the **Proposed** method show that, although different source embeddings are important to different degrees, by using all source embeddings we can obtain the best results. Different source embeddings are trained from different resources and by optimising different objectives. Therefore, for different words, the local neighbours predicted by different source embeddings will be complementary. Unlike the other methods, the **Proposed** method never compares different source embeddings' vectors directly, but only via the neighbourhood reconstruction weights. Consequently, the **Proposed** method is unaffected by relative weighting of source embeddings. In contrast, the **CONC** is highly sensitive against the weighting. In fact, we confirmed that the performance scores of the CONC method were decreased by 3–10 points when we did not do the weight tuning described in Section 4.2. The unnecessity of the weight tuning is thus a clear advantage of the Proposed method.

To investigate the effect of the dimensionality $d^{\mathcal{P}}$ on the meta-embeddings learnt by the proposed method, in Figure 1a, we fix the neighbourhood size $N = 1200$ and measure the performance on semantic similarity measurement tasks when varying $d^{\mathcal{P}}$. Overall, we see that the performance peaks around $d^{\mathcal{P}} = 300$. Such behaviour can be explained by the fact that smaller $d^{\mathcal{P}}$ dimensions are unable to preserve information contained in the source embeddings, whereas increasing $d^{\mathcal{P}}$ beyond the rank of the weight matrix $\mathbf{W}$ is likely to generate noisy eigenvectors.

In Figure 1b, we study the effect of increasing the neighbourhood size n equally for all words in all source embeddings, while fixing the dimensionality of the meta-embedding $d^{\mathcal{P}} = 300$. Initially, performance increases with the neighbourhood size and then saturates. This implies that in practice a small local neighbourhood is adequate to capture the differences in source embeddings.

4.5 Complementarity of Resources

We have shown empirically in Section 4.4 that using the proposed method it is possible to obtain superior meta-embeddings from a diverse set of source embeddings. One important scenario where meta-embedding could be potentially useful is when the source embeddings are trained on different complementary resources, where each resource share little common vocabulary. For example, one source embedding might have been trained on Wikipedia whereas a second source embedding might have been trained on tweets.

To evaluate the effectiveness of the proposed meta-embedding learning method under such settings, we design the following experiment. We select **MEN** dataset, the largest among all semantic similarity benchmarks, which contains 751 unique words in 3000 human-rated word-pairs for semantic similarity. Next, we randomly split the set of words into two sets with different overlap ratios. We then select sentences from 2017 January dump of Wikipedia that contains words from only one of the two sets. We create two corpora of roughly equal number of sentences via this procedure for different overlap ratios. We train skip-gram with negative sampling (SGNS) (Mikolov, Chen, and Dean, 2013) on one corpus to create source embedding S_1 and GloVe (Pennington, Socher, and Manning, 2014) on the other corpus to create source embedding S_2. Finally, we use the proposed method to meta-embed S_1 and S_2.

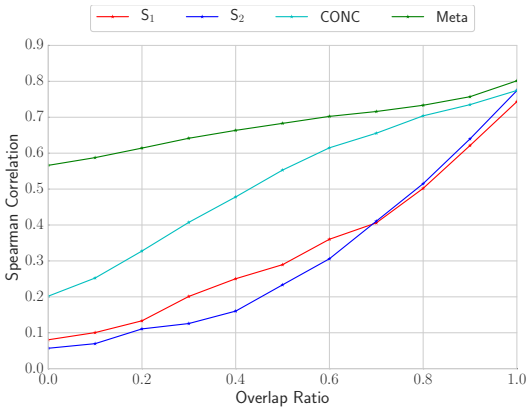


Figure 2: Two word embedding learning algorithms trained on different but overlapping corpora to produce two source embeddings S_1 and S_2. and their meta-embedding.

Figure 2 shows the Spearman correlation between the human similarity ratings and cosine similarities computed using the word embeddings on the **MEN** dataset for S_1, S_2 and their meta-embeddings created using the proposed method (Meta) and concatenation baseline (CONC). From Figure 2, we see that the meta embeddings obtain the best performance across all overlap ratios. The improvements are larger when the overlap between the corpora is smaller, and diminishes when the two corpora becomes identical. This result shows that our proposed meta-embedding learning method captures the complementary information available in different source embeddings to create more accurate word embeddings. Moreover, it shows that by considering the local neighbourhoods in each of the source embeddings separately, we can obviate the need to predict embeddings for missing words in a particular source embedding, which was a limitation in the method proposed by Yin and Schütze (2016).

5 Conclusion

We proposed an unsupervised locally linear method for learning meta-embeddings from a given set of pre-trained source embeddings. Experiments on several NLP tasks show the accuracy of the proposed method, which outperforms previously proposed meta-embedding learning methods on multiple benchmark datasets. In future, we plan to extend the proposed method to learn cross-lingual meta-embeddings by incorporating both cross-lingual as well as monolingual information.

References

2000. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for the Industrial and Applied Mathematics.

Bansal, M.; Gimpel, K.; and Livescu, K. 2014. Tailoring continouos word representations for dependency parsing. In *Proc. of ACL*.

Baroni, M.; Dinu, G.; and Kruszewski, G. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proc. of ACL*, 238–247.

Bruni, E.; Boleda, G.; Baroni, M.; and Tran, N. K. 2012. Distributional semantics in technicolor. In *Proc. of ACL*, 136–145.

Chen, Y.; Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2013. The expressive power of word embeddings. In *Proc. of ICML Workshop*.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*, 160 – 167.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Goikoetxea, J.; Agirre, E.; and Soroa, A. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *Proc. of AAAI*, 2608–2614.

Halko, N.; Martinsson, P. G.; and Tropp, J. A. 2010. Finding structure with randomness: Probabilistic algorithms for constructung approximate matrix decompositions. *SIAM REVIEW* 53(2):217 – 288.

Hill, F.; Cho, K.; Jean, S.; Devin, C.; and Bengio, Y. 2014. Not all neural embeddings are born equal. In *NIPS workshop*.

Hill, F.; Cho, K.; Jean, S.; Devin, C.; and Bengio, Y. 2015. Embedding word similarity with neural machine translation. In *ICLR Workshop*.

Hill, F.; Reichart, R.; and Korhonen, A. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4):665–695.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *Proc. of ACL*, 873–882.

Jurgens, D. A.; Mohammad, S.; Turney, P. D.; and Holyoak, K. J. 2012. Measuring degrees of relational similarity. In *Proc. of SemEval*.

Kibriya, A. M., and Frank, E. 2007. An empirical comparison of exact nearest neighbout algorithms. In *Proc. of PKDD*, 140–151.

Luo, Y.; Tang, J.; Yan, J.; Xu, C.; and Chen, Z. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Proc. of AAAI*, 1982–1988.

Luong, M.-T.; Socher, R.; and Manning, C. D. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, 3111–3119.

Mikolov, T.; Chen, K.; and Dean, J. 2013. Efficient estimation of word representation in vector space. In *Proc. of ICLR*.

Miller, G., and Charles, W. 1998. Contextual correlates of semantic similarity. *Language and Cognitive Processes* 6(1):1–28.

Mnih, A., and Hinton, G. E. 2009. A scalable hierarchical distributed language model. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Proc. of NIPS*. 1081–1088.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, 115–124.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: global vectors for word representation. In *Proc. of EMNLP*, 1532–1543.

Rubenstein, H., and Goodenough, J. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8:627–633.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, 1631–1642.

Tian, Y.; Kullkarni, V.; Perozzi, B.; and Skiena, S. 2016. On the convergent properties of word embedding methods. *arXiv*.

Tsuboi, Y. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proc. of EMNLP*, 938–950.

Turian, J.; Ratinov, L.; and Bengio, Y. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*, 384 – 394.

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

Vylomova, E.; Rimell, L.; Cohn, T.; and Baldwin, T. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relational learning. In *ACL*, 1671–1682.

Yin, W., and Schütze, H. 2016. Learning meta-embeddings by using ensembles of embedding sets. In *Proc. of ACL*, 1351–1360.

Zou, W. Y.; Socher, R.; Cer, D.; and Manning, C. D. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of EMNLP*, 1393–1398.