

# Learning Meta-Embeddings by Using Ensembles of Embedding Sets

Wenpeng Yin and Hinrich Schütze

Center for Information and Language Processing  
University of Munich, Germany  
wenpeng@cis.uni-muenchen.de

## Abstract

Word embeddings – distributed representations of words – in deep learning are beneficial for many tasks in natural language processing (NLP). However, different embedding sets vary greatly in quality and characteristics of the captured semantics. Instead of relying on a more advanced algorithm for embedding learning, this paper proposes an ensemble approach of combining different public embedding sets with the aim of learning meta-embeddings. Experiments on word similarity and analogy tasks and on part-of-speech tagging show better performance of meta-embeddings compared to individual embedding sets. One advantage of meta-embeddings is the increased vocabulary coverage. We will release our meta-embeddings publicly.

## 1 Introduction

Recently, deep neural network (NN) models have achieved remarkable results in NLP (Collobert and Weston, 2008; Sutskever et al., 2014; Rocktäschel et al., 2015). One reason for these results are word embeddings, compact distributed word representations learned in an unsupervised manner from large corpora (Bengio et al., 2003; Mnih and Hinton, 2009; Mikolov et al., 2010; Mikolov, 2012; Mikolov et al., 2013a).

Some prior work has studied differences in performance of different embedding sets. For example, Chen et al. (2013) showed that the embedding sets HLBL (Mnih and Hinton, 2009), SENNA (Collobert and Weston, 2008), Turian (Turian et al., 2010) and Huang (Huang et al., 2012) have great

variance in quality and characteristics of the semantics captured. Hill et al. (2014; 2015a) showed that embeddings learned by NN machine translation models can outperform three representative monolingual embedding sets: word2vec (Mikolov et al., 2013b), GloVe (Pennington et al., 2014) and CW (Collobert and Weston, 2008). Bansal et al. (2014) found that Brown clustering, SENNA, CW, Huang and word2vec yield significant gains for dependency parsing. Moreover, using these representations together achieved the best results, suggesting their complementarity. These prior studies motivate us to explore an ensemble approach. Since each embedding set is trained by a different NN on a different corpus and can be treated as a distinct description of words, our expectation is that the ensemble contains more information than each component embedding set. We want to leverage this *diversity* to learn better-performing word embeddings.

The ensemble approach has two benefits. First, *enhancement* of the representations: meta-embeddings perform better than the individual embedding sets. Second, *coverage*: meta-embeddings cover more words than the individual embedding sets. The first three ensemble methods we introduce are CONC, SVD and 1TON and they directly only have the benefit of enhancement. They learn meta-embeddings on the overlapping vocabulary of the embedding sets. CONC concatenates the vectors of a word from the different embedding sets. SVD performs dimension reduction on this concatenation. 1TON assumes that a meta-embedding for the word exists and uses this meta-embedding to predict representations of the word in the indi-

vidual embedding sets – the resulting fine-tuned meta-embedding is expected to contain knowledge from all individual embedding sets.

To also address the objective of increased coverage of the vocabulary, we introduce 1TON<sup>+</sup>, a modification of 1TON that learns meta-embeddings for all words in the *vocabulary union* in one step. Let an out-of-vocabulary (OOV) word  $w$  of embedding set ES be a word that is not covered by ES (i.e., ES does not contain an embedding for  $w$ ).<sup>1</sup> 1TON<sup>+</sup> first randomly initializes the embeddings for OOVs and the meta-embeddings, then uses a prediction setup similar to 1TON to update meta-embeddings *as well as OOV embeddings*. Thus, 1TON<sup>+</sup> simultaneously achieves two goals: learning meta-embeddings and extending the vocabulary (for both meta-embeddings and individual embedding sets).

An alternative method that increases coverage is **MUTUALLEARNING**. **MUTUALLEARNING** learns the embedding for a word that is an OOV in embedding set from its embeddings in other embedding sets. We will use **MUTUALLEARNING** to increase coverage for CONC, SVD and 1TON, so that these three methods (when used together with **MUTUALLEARNING**) have the advantages of both performance enhancement and increased coverage.

In summary, meta-embeddings have two benefits compared to individual embedding sets: *enhancement of performance and improved coverage of the vocabulary*. Below, we demonstrate this experimentally for three tasks: word similarity, word analogy and POS tagging.

If we simply view meta-embeddings as a way of coming up with better embeddings, then the alternative is to develop a single embedding learning algorithm that produces better embeddings. Some improvements proposed before have the disadvantage of increasing the training time of embedding learning substantially; e.g., the NNLM presented in (Bengio et al., 2003) is an order of magnitude less efficient than an algorithm like word2vec and, more generally, replacing a linear objective function with a nonlinear objective function increases training time. Similarly, fine-tuning the hyperparameters of the embedding learning algorithm is complex and

time consuming. In many cases, it is not possible to retrain using a different algorithm because the corpus is not publicly available. But even if these obstacles could be overcome, it is unlikely that there ever will be a single “best” embedding learning algorithm. So the current situation of multiple embedding sets with different properties being available is likely to persist for the foreseeable future. Meta-embedding learning is a simple and efficient way of taking advantage of this diversity. As we will show below they combine several complementary embedding sets and the resulting meta-embeddings are stronger than each individual set.

## 2 Related Work

Related work has focused on improving performance on specific tasks by using several embedding sets simultaneously. To our knowledge, there is no work that aims to learn generally useful meta-embeddings from individual embedding sets.

Tsuboi (2014) incorporated word2vec and GloVe embeddings into a POS tagging system and found that using these two embedding sets together was better than using them individually. Similarly, Turian et al. (2010) found that using Brown clusters, CW embeddings and HLBL embeddings for NER and chunking tasks together gave better performance than using these representations individually.

Luo et al. (2014) adapted CBOW (Mikolov et al., 2013a) to train word embeddings on different datasets – a Wikipedia corpus, search click-through data and user query data – for web search ranking and for word similarity. They showed that using these embeddings together gives stronger results than using them individually.

These papers show that using multiple embedding sets is beneficial. However, they either use embedding sets trained on the same corpus (Turian et al., 2010) or enhance embedding sets by more training data, not by innovative learning algorithms (Luo et al., 2014). In our work, we can leverage any publicly available embedding set learned by any learning algorithm. Our meta-embeddings are generically useful and are learned by supervised training of an *explicit model of the dependencies* between embedding sets and (except for CONC) not by *simple concatenation*.

<sup>1</sup>We do not consider words in this paper that are not covered by any of the individual embedding sets. OOV always refers to a word that is covered by at least one embedding set.

### 3 Experimental Embedding Sets

In this work, we use five released embedding sets. (i) **HLBL**. Hierarchical log-bilinear (Mnih and Hinton, 2009) embeddings released by Turian et al. (2010);<sup>2</sup> 246,122 word embeddings, 100 dimensions; training corpus: RCV1 corpus (Reuters English newswire, August 1996 – August 1997). (ii) **Huang**.<sup>3</sup> Huang et al. (2012) incorporated global context to deal with challenges raised by words with multiple meanings; 100,232 word embeddings, 50 dimensions; training corpus: April 2010 snapshot of Wikipedia. (iii) **GloVe**.<sup>4</sup> (Pennington et al., 2014). 1,193,514 word embeddings, 300 dimensions; training corpus: 42 billion tokens of web data, from Common Crawl. (iv) **CW** (Collobert and Weston, 2008). Released by Turian et al. (2010);<sup>5</sup> 268,810 word embeddings, 200 dimensions; training corpus: same as HLBL. (v) **word2vec** (Mikolov et al., 2013b) CBOW;<sup>6</sup> 929,022 word embeddings (we discard phrase embeddings), 300 dimensions; training corpus: Google News (about 100 billion words).

The intersection of the five vocabularies has size 35,965, the union has size 2,788,636.

### 4 Ensemble Methods

This section introduces the four ensemble methods: CONC, SVD, 1TON and 1TON<sup>+</sup>.

#### 4.1 CONC: Concatenation

In **CONC**, the meta-embedding of  $w$  is the concatenation of five embeddings, one each from the five embedding sets. For GloVe, we perform L2 normalization for each dimension across the vocabulary as recommended by the GloVe authors. Then each embedding of each embedding set is L2-normalized. This ensures that each embedding set contributes equally (a value between -1 and 1) when we compute similarity via dot product.

We would like to make use of prior knowledge and give more weight to well performing embedding sets. In this work, we give GloVe and word2vec

<sup>2</sup><http://metaoptimize.com/projects/wordreprs/>

<sup>3</sup><http://ai.stanford.edu/~ehhuang/>

<sup>4</sup><http://nlp.stanford.edu/projects/glove/>

<sup>5</sup><http://metaoptimize.com/projects/wordreprs/>

<sup>6</sup><http://code.google.com/p/Word2Vec/>

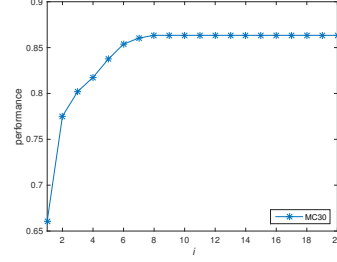


Figure 1: Performance vs. Weight scalar  $i$

weight  $i > 1$  and weight 1 to the other three embedding sets. We use MC30 (Miller and Charles, 1991) as dev set, since all embedding sets fully cover it. We set  $i = 8$ , the value in Figure 1 where performance reaches a plateau. After L2 normalization, GloVe and word2vec embeddings are multiplied by  $i$  and remaining embedding sets are left unchanged.

The dimensionality of CONC meta-embeddings is  $k = 100 + 50 + 300 + 200 + 300 = 950$ .

#### 4.2 SVD: Singular Value Decomposition

We do SVD on above weighted concatenation vectors of dimension  $k = 950$ .

Given a set of CONC representations for  $n$  words, each of dimensionality  $k$ , we compute an SVD decomposition  $C = USV^T$  of the corresponding  $n \times k$  matrix  $C$ . We then use  $U_d$ , the first  $d$  dimensions of  $U$ , as the SVD meta-embeddings of the  $n$  words. We apply L2-normalization to embeddings; similarities of SVD vectors are computed as dot products.

$d$  denotes the dimensionality of meta-embeddings in SVD, 1TON and 1TON<sup>+</sup>. We use  $d = 200$  throughout and investigate the impact of  $d$  below.

#### 4.3 1TON

Figure 2 depicts the simple neural network we employ to learn meta-embeddings in 1TON. White rectangles denote known embeddings. The target to learn is the meta-embedding (shown as shaded rectangle). Meta-embeddings are initialized randomly.

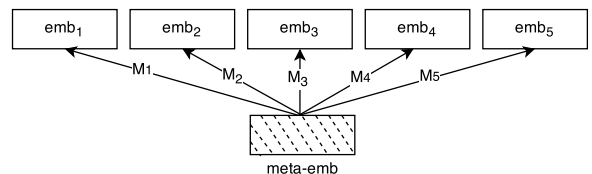


Figure 2: 1ton

Let  $c$  be the number of embedding sets under consideration,  $V_1, V_2, \dots, V_i, \dots, V_c$  their vocabularies and  $V^\cap = \bigcap_{i=1}^c V_i$  the intersection, used as training set. Let  $V_*$  denote the meta-embedding space. We define a projection  $f_{*i}$  from space  $V_*$  to space  $V_i$  ( $i = 1, 2, \dots, c$ ) as follows:

$$\hat{\mathbf{w}}_i = \mathbf{M}_{*i} \mathbf{w}_* \quad (1)$$

where  $\mathbf{M}_{*i} \in \mathbb{R}^{d_i \times d}$ ,  $\mathbf{w}_* \in \mathbb{R}^d$  is the meta-embedding of word  $w$  in space  $V_*$  and  $\hat{\mathbf{w}}_i \in \mathbb{R}^{d_i}$  is the projected (or learned) representation of word  $w$  in space  $V_i$ . The training objective is minimizing the sum of (i) squared error:

$$E = \sum_i |\hat{\mathbf{w}}_i - \mathbf{w}_i|^2 \quad (2)$$

and (ii) L2 cost (sum of squares) of the projection weights  $\mathbf{M}_{*i}$ .

As for CONC and SVD, we weight GloVe and word2vec by  $i = 8$ . For 1TON, we implement this by applying the factor  $i$  to the corresponding loss part of the squared error.

The principle of 1TON is that we treat each individual embedding as a projection of the meta-embedding, similar to principal component analysis. An embedding is a description of the word based on the corpus and the model that were used to create it. The meta-embedding tries to recover a more comprehensive description of the word when it is trained to predict the individual descriptions.

1TON can also be understood as a sentence modeling process, similar to DBOW (Le and Mikolov, 2014). The embedding of each word in a sentence  $s$  is a partial description of  $s$ . DBOW combines all partial descriptions to form a comprehensive description of  $s$ . DBOW initializes the sentence representation randomly, then uses this representation to predict the representations of individual words. The sentence representation of  $s$  corresponds to the meta-embedding in 1TON; and the representations of the words in  $s$  correspond to the five embeddings for a word in 1TON.

#### 4.4 1TON<sup>+</sup>

Recall that an OOV (with respect to embedding set ES) is defined as a word unknown in ES. 1TON<sup>+</sup> is an extension of 1TON that learns embeddings for

OOVs; thus, it does not have the limitation that it can only be run on overlapping vocabulary.

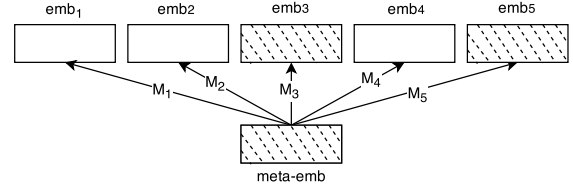


Figure 3: 1TON<sup>+</sup>

Figure 3 depicts 1TON<sup>+</sup>. In contrast to Figure 2, we assume that the current word is an OOV in embedding sets 3 and 5. Hence, in the new learning task, embeddings 1, 2, 4 are known, and embeddings 3 and 5 and the meta-embedding are targets to learn.

We initialize all OOV representations and meta-embeddings randomly and use the same mapping formula as for 1TON to connect a meta-embedding with the individual embeddings. Both meta-embedding and initialized OOV embeddings are updated during training.

Each embedding set contains information about only a part of the overall vocabulary. However, it can predict what the remaining part should look like by comparing words it knows with the information other embedding sets provide about these words. Thus, 1TON<sup>+</sup> learns a model of the dependencies between the individual embedding sets and can use these dependencies to infer what the embedding of an OOV should look like.

CONC, SVD and 1TON compute meta-embeddings only for the intersection vocabulary. 1TON<sup>+</sup> computes meta-embeddings for the union of all individual vocabularies, thus greatly increasing the coverage of individual embedding sets.

## 5 MUTUALLEARNING

MUTUALLEARNING is a method that extends CONC, SVD and 1TON such that they have increased coverage of the vocabulary. With MUTUALLEARNING, all four ensemble methods – CONC, SVD, 1TON and 1TON<sup>+</sup> – have the benefits of both performance enhancement and increased coverage and we can use criteria like performance, compactness and efficiency of training to select the best ensemble method for a particular application.

	bs	lr	L2 weight
1TON	200	0.005	$5 \times 10^{-4}$
MUTUALLEARNING (ml)	200	0.01	$5 \times 10^{-8}$
1TON <sup>+</sup>	2000	0.005	$5 \times 10^{-4}$

**Table 1:** Training setup. bs: batch size; lr: learning rate.

MUTUALLEARNING is applied to learn OOV embeddings for all  $c$  embedding sets; however, for ease of exposition, let us assume we want to compute embeddings for OOVs for embedding set  $j$  only, based on known embeddings in the other  $c - 1$  embedding sets, with indexes  $i \in \{1 \dots j - 1, j + 1 \dots c\}$ . We do this by learning  $c - 1$  mappings  $f_{ij}$ , each a projection from embedding set  $E_i$  to embedding set  $E_j$ .

Similar to Section 4.3, we train mapping  $f_{ij}$  on the intersection  $V_i \cap V_j$  of the vocabularies covered by the two embedding sets. Formally,  $\hat{\mathbf{w}}_j = f_{ij}(\mathbf{w}_i) = \mathbf{M}_{ij}\mathbf{w}_i$  where  $\mathbf{M}_{ij} \in \mathbb{R}^{d_j \times d_i}$ ,  $\mathbf{w}_i \in \mathbb{R}^{d_i}$  denotes the representation of word  $w$  in space  $V_i$  and  $\hat{\mathbf{w}}_j$  is the projected meta-embedding of word  $w$  in space  $V_j$ . Training loss has the same form as for 1TON. A total of  $c - 1$  projections  $f_{ij}$  are trained to learn OOV embeddings for embedding set  $j$ .

Let  $w$  be a word unknown in the vocabulary  $V_j$  of embedding set  $j$ , but known in  $V_1, V_2, \dots, V_k$ . To compute an embedding for  $w$  in  $V_j$ , we first compute the  $k$  projections  $f_{1j}(\mathbf{w}_1), f_{2j}(\mathbf{w}_2), \dots, f_{kj}(\mathbf{w}_k)$  from the source spaces  $V_1, V_2, \dots, V_k$  to the target space  $V_j$ . Then, the element-wise average of  $f_{1j}(\mathbf{w}_1), f_{2j}(\mathbf{w}_2), \dots, f_{kj}(\mathbf{w}_k)$  is treated as the representation of  $w$  in  $V_j$ . Our motivation is that – assuming there is a true representation of  $w$  in  $V_j$  and assuming the projections were learned well – we would expect all the projected vectors to be close to the true representation. Also, each source space contributes potentially complementary information. Hence averaging them is a balance of knowledge from all source spaces.

## 6 Experiments

We train NNs by back-propagation with AdaGrad (Duchi et al., 2011) and mini-batches. Table 1 gives hyperparameters.

We report results on three tasks: word similarity, word analogy and POS tagging.

### 6.1 Word Similarity and Analogy Tasks

We evaluate on SimLex-999 (Hill et al., 2015b), WordSim353 (Finkelstein et al., 2001), RG (Rubenstein and Goodenough, 1965) and RW (Luong et al., 2013). For completeness, we also show results for MC30, the validation set.

The word analogy task proposed in (Mikolov et al., 2013b) consists of questions like, “ $a$  is to  $b$  as  $c$  is to  $_$ ?”. The dataset contains 19,544 such questions, divided into a semantic subset of size 8869 and a syntactic subset of size 10,675.

Table 2 gives **results on similarity and analogy**. Numbers in parentheses are line numbers in what follows. Block “ind-full” (1-5) lists the performance of individual embedding sets on the *full vocabulary*. Results on lines 6-34 are for the intersection of the vocabularies of the five embedding sets: “ind-overlap” contains the performance of individual embedding sets, “ensemble” the performance of our four ensemble methods and “discard” the performance when one component set is removed.

The four ensemble approaches are very promising (31-34). For CONC, discarding HLBL, Huang or CW does not hurt performance: CONC (31), CONC(-HLBL) (11), CONC(-Huang) (12) and CONC(-CW) (14) beat each individual embedding set (6-10) in all tasks. GloVe contributes most in SimLex-999, WS353, MC30 and RG; word2vec contributes most in RW and word analogy tasks.

SVD (32) reduces the dimensionality of CONC from 950 to 200, but still gains performance in SimLex-999 and RG. GloVe contributes most in SVD (larger losses on line 18 vs. lines 16-17, 19-20). Other embeddings contribute inconsistently.

1TON performs well only on word analogy, but it gains great improvement when discarding CW embeddings (24). 1TON<sup>+</sup> performs better than 1TON: it has stronger results when considering all embedding sets, and can still outperform individual embedding sets while discarding HLBL (26), Huang (27) or CW (29).

These results demonstrate that ensemble methods using multiple embedding sets produce stronger embeddings. However, it does not mean the more embedding sets the better. Whether an embedding set helps, depends on the complementarity among the sets as well as how we measure the ensemble results.

	Model	SL999	WS353	MC30	RG	RW	sem.	syn.	tot.
ind-full	1 HLBL	22.1 (1)	35.7 (3)	41.5 (0)	35.2 (1)	19.1 (892)	27.1 (423)	22.8 (198)	24.7
	2 Huang	9.7 (3)	61.7 (18)	65.9 (0)	63.0 (0)	6.4 (982)	8.4 (1016)	11.9 (326)	10.4
	3 GloVe	45.3 (0)	75.4 (18)	83.6 (0)	82.9 (0)	48.7 (21)	81.4 (0)	70.1 (0)	75.2
	4 CW	15.6 (1)	28.4 (3)	21.7 (0)	29.9 (1)	15.3 (896)	17.4 (423)	5.0 (198)	10.5
	5 W2V	44.2 (0)	69.8 (0)	78.9 (0)	76.1 (0)	53.4 (209)	77.1 (0)	74.4 (0)	75.6
ind-overlap	6 HLBL	22.3 (3)	34.8 (21)	41.5 (0)	35.2 (1)	22.2 (1212)	13.8 (8486)	15.4 (1859)	15.4
	7 Huang	9.7 (3)	62.0 (21)	65.9 (0)	64.1 (1)	3.9 (1212)	27.9 (8486)	9.9 (1859)	10.7
	8 GloVe	45.0 (3)	75.5 (21)	83.6 (0)	82.4 (1)	59.1 (1212)	91.1 (8486)	68.2 (1859)	69.2
	9 CW	16.0 (3)	30.8 (21)	21.7 (0)	29.9 (1)	17.4 (1212)	11.2 (8486)	2.3 (1859)	2.7
	10 W2V	44.1 (3)	69.3 (21)	78.9 (0)	75.4 (1)	61.5 (1212)	89.3 (8486)	72.6 (1859)	73.3
discard	11 CONC (-HLBL)	<b>46.0</b> (3)	<b>76.5</b> (21)	<b>86.3</b> (0)	<b>82.5</b> (1)	<b>63.0</b> (1211)	<b>93.2</b> (8486)	<b>74.0</b> (1859)	<b>74.8</b>
	12 CONC (-Huang)	<b>46.1</b> (3)	<b>76.5</b> (21)	<b>86.3</b> (0)	<b>82.5</b> (1)	<b>62.9</b> (1212)	<b>93.2</b> (8486)	<b>74.0</b> (1859)	<b>74.8</b>
	13 CONC (-GloVe)	44.0 (3)	69.4 (21)	79.1 (0)	<b>75.6</b> (1)	61.5 (1212)	89.3 (8486)	<b>72.7</b> (1859)	<b>73.4</b>
	14 CONC (-CW)	<b>46.0</b> (3)	<b>76.5</b> (21)	<b>86.6</b> (0)	<b>82.5</b> (1)	<b>62.9</b> (1212)	<b>93.2</b> (8486)	<b>73.9</b> (1859)	<b>74.7</b>
	15 CONC (-W2V)	45.0 (3)	75.5 (21)	83.6 (0)	82.4 (1)	59.1 (1212)	90.9 (8486)	68.3 (1859)	69.2
	16 SVD (-HLBL)	<b>48.5</b> (3)	<b>76.1</b> (21)	<b>85.6</b> (0)	<b>82.7</b> (1)	61.5 (1211)	90.6 (8486)	69.5 (1859)	70.4
	17 SVD (-Huang)	<b>48.8</b> (3)	<b>76.5</b> (21)	<b>85.4</b> (0)	<b>83.5</b> (1)	<b>61.7</b> (1212)	<b>91.4</b> (8486)	69.8 (1859)	70.7
	18 SVD (-GloVe)	<b>46.2</b> (3)	66.9 (21)	81.6 (0)	78.6 (1)	59.1 (1212)	88.8 (8486)	67.3 (1859)	68.2
	19 SVD (-CW)	<b>48.5</b> (3)	<b>76.1</b> (21)	<b>85.7</b> (0)	<b>82.7</b> (1)	61.5 (1212)	90.6 (8486)	69.5 (1859)	70.4
	20 SVD (-W2V)	<b>49.4</b> (3)	<b>79.0</b> (21)	<b>87.3</b> (0)	80.7 (1)	59.1 (1212)	90.3 (8486)	66.0 (1859)	67.1
	21 1ToN (-HLBL)	<b>46.3</b> (3)	75.5 (21)	82.4 (0)	81.0 (1)	60.1 (1211)	<b>91.9</b> (8486)	<b>75.9</b> (1859)	<b>76.5</b>
	22 1ToN (-Huang)	<b>46.5</b> (3)	75.4 (21)	82.4 (0)	82.3 (1)	60.2 (1212)	<b>93.5</b> (8486)	<b>76.3</b> (1859)	<b>77.0</b>
	23 1ToN (-GloVe)	43.4 (3)	66.5 (21)	76.5 (0)	75.3 (1)	56.5 (1212)	89.0 (8486)	<b>73.8</b> (1859)	<b>74.5</b>
	24 1ToN (-CW)	<b>47.4</b> (3)	<b>76.5</b> (21)	<b>84.8</b> (0)	<b>83.4</b> (1)	<b>62.0</b> (1212)	<b>91.4</b> (8486)	<b>73.1</b> (1859)	<b>73.8</b>
	25 1ToN (-W2V)	<b>46.3</b> (3)	<b>75.9</b> (21)	80.1 (0)	78.6 (1)	56.8 (1212)	<b>92.2</b> (8486)	72.2 (1859)	73.0
	26 1ToN <sup>+</sup> (-HLBL)	<b>46.1</b> (3)	<b>75.8</b> (21)	<b>85.5</b> (0)	<b>83.3</b> (1)	<b>62.3</b> (1211)	<b>92.2</b> (8486)	<b>76.2</b> (1859)	<b>76.9</b>
	27 1ToN <sup>+</sup> (-Huang)	<b>46.2</b> (3)	<b>76.1</b> (21)	<b>86.3</b> (0)	<b>83.3</b> (1)	<b>62.2</b> (1212)	<b>93.8</b> (8486)	<b>76.1</b> (1859)	<b>76.8</b>
	28 1ToN <sup>+</sup> (-GloVe)	<b>45.3</b> (3)	71.2 (21)	80.0 (0)	75.7 (1)	<b>62.5</b> (1212)	90.0 (8486)	<b>73.3</b> (1859)	<b>74.0</b>
	29 1ToN <sup>+</sup> (-CW)	<b>46.9</b> (3)	<b>78.1</b> (21)	<b>85.5</b> (0)	<b>83.9</b> (1)	<b>62.7</b> (1212)	<b>91.8</b> (8486)	<b>73.3</b> (1859)	<b>74.1</b>
	30 1ToN <sup>+</sup> (-W2V)	<b>45.8</b> (3)	<b>76.2</b> (21)	<b>84.4</b> (0)	<b>83.1</b> (1)	60.9 (1212)	<b>92.4</b> (8486)	72.4 (1859)	73.2
ensemble	31 CONC	<b>46.0</b> (3)	<b>76.5</b> (21)	<b>86.3</b> (0)	<b>82.5</b> (1)	<b>62.9</b> (1212)	<b>93.2</b> (8486)	<b>74.0</b> (1859)	<b>74.8</b>
	32 SVD	<b>48.5</b> (3)	<b>76.0</b> (21)	<b>85.7</b> (0)	<b>82.7</b> (1)	61.5 (1212)	90.6 (8486)	69.5 (1859)	70.4
	33 1ToN	<b>46.4</b> (3)	74.5 (21)	80.7 (0)	80.7 (1)	60.1 (1212)	<b>91.9</b> (8486)	<b>76.1</b> (1859)	<b>76.8</b>
	34 1ToN <sup>+</sup>	<b>46.3</b> (3)	75.3 (21)	<b>85.2</b> (0)	<b>82.7</b> (1)	<b>61.6</b> (1212)	<b>92.5</b> (8486)	<b>76.3</b> (1859)	<b>77.0</b>

**Table 2:** Results on five word similarity tasks and analogical reasoning. The number of OOVs is given in parentheses for each result. “ind-full/ind-overlap”: individual embedding sets with respective full/overlapping vocabulary; “ensemble”: ensemble results using all five embedding sets; “discard”: one of the five embedding sets is removed. If a result is better than all methods in “ind-overlap”, then it is bolded.

		RW(21)				semantic				syntactic				total			
		RND	AVG	ml	1ToN <sup>+</sup>	RND	AVG	ml	1ToN <sup>+</sup>	RND	AVG	ml	1ToN <sup>+</sup>	RND	AVG	ml	1ToN <sup>+</sup>
ind	HLBL	7.4	6.9	17.3	17.5	26.3	26.4	26.3	26.4	22.4	22.4	22.7	22.9	24.1	24.2	24.4	24.5
	Huang	4.4	4.3	6.4	6.4	1.2	2.7	21.8	22.0	7.7	4.1	10.9	11.4	4.8	3.3	15.8	16.2
	CW	7.1	10.6	17.3	17.7	17.2	17.2	16.7	18.4	4.9	5.0	5.0	5.5	10.5	10.5	10.3	11.4
ensemble	CONC	14.2	16.5	48.3	–	4.6	18.0	88.1	–	62.4	15.1	74.9	–	36.2	16.3	81.0	–
	SVD	12.4	15.7	47.9	–	4.1	17.5	87.3	–	54.3	13.6	70.1	–	31.5	15.4	77.9	–
	1ToN	16.7	11.7	48.5	–	4.2	17.6	88.2	–	60.0	15.0	76.8	–	34.7	16.1	82.0	–
	1ToN <sup>+</sup>	–	–	–	48.8	–	–	–	88.4	–	–	–	76.3	–	–	–	81.1

**Table 3:** Comparison of effectiveness of four methods for learning OOV embeddings. RND: random initialization. AVG: average of embeddings of known words. ml: MUTUALLEARNING. RW(21) means there are still 21 OOVs for the vocabulary union.



CONC, the simplest ensemble, has robust performance. However, using embeddings of size 950 as input may mean too many parameters to tune for deep learning. The other three methods – SVD, 1TON, 1TON<sup>+</sup> – all have the advantage of smaller dimensionality. SVD reduces CONC’s dimensionality dramatically and still keeps competitive performance, especially on word similarity. 1TON is competitive on analogy, but weak on word similarity. 1TON<sup>+</sup> performs consistently strongly on word similarity and analogy.

Not all state-of-the-art results are included in Table 2. One reason is that a fair comparison is only possible on the shared vocabulary, so methods without released embeddings cannot be included. However, GloVe and word2vec are widely recognized as state-of-the-art embeddings. In any case, our main contribution is to present ensemble frameworks which show that a combination of complementary embedding sets produces better-performing meta-embeddings.

**System comparison of learning OOV embeddings.** In Table 3, we extend the vocabularies of each individual embedding set (“ind” block) and our ensemble approaches (“ensemble” block) to the vocabulary union, reporting results on RW and analogy – these tasks contain the most OOVs. As both word2vec and GloVe have full coverage on analogy, we do not rereport them in this table. For each embedding set, we can compute the representation of an OOV (i) as a randomly initialized vector (RND); (ii) as the average of embeddings of all known words (AVG); (iii) by MUTUALLEARNING (ml) and (iv) by 1TON<sup>+</sup>. 1TON<sup>+</sup> learns OOV embeddings for individual embedding sets and meta-embeddings simultaneously, and it would not make sense to replace these OOV embeddings computed by 1TON<sup>+</sup> with embeddings computed by “RND/AVG/ml”. Hence, we do not report “RND/AVG/ml” results for 1TON<sup>+</sup>.

Table 3 shows four interesting aspects. (i) MUTUALLEARNING helps much if an embedding set has lots of OOVs in certain task; e.g., MUTUALLEARNING is much better than AVG and RND on RW, and outperforms RND considerably for CONC, SVD and 1TON on analogy. However, it cannot make big difference for HLBL/CW on analogy, probably because these two embedding sets have much fewer

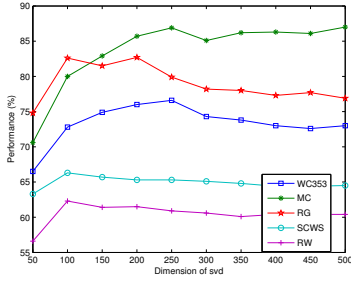
OOVs, in which case AVG and RND work well enough. (ii) AVG produces bad results for CONC, SVD and 1TON on analogy, especially in the syntactic subtask. We notice that those systems have large numbers of OOVs in word analogy task. If for analogy “*a* is to *b* as *c* is to *d*”, all four of *a*, *b*, *c*, *d* are OOVs, then they are represented with the same average vector. Hence, similarity between  $b - a + c$  and each OOV is 1.0. In this case, it is almost impossible to predict the correct answer *d*. Unfortunately, methods CONC, SVD and 1TON have many OOVs, resulting in the low numbers in Table 3. (iii) MUTUALLEARNING learns very effective embeddings for OOVs. CONC-ml, 1TON-ml and SVD-ml all get better results than word2vec and GloVe on analogy (e.g., for semantic analogy: 88.1, 87.3, 88.2 vs. 81.4 for GloVe). Considering further their bigger vocabulary, these ensemble methods are very strong representation learning algorithms. (iv) The performance of 1TON<sup>+</sup> for learning embeddings for OOVs is competitive with MUTUALLEARNING. For HLBL/Huang/CW, 1TON<sup>+</sup> performs slightly better than MUTUALLEARNING in all four metrics. Comparing 1TON-ml with 1TON<sup>+</sup>, 1TON<sup>+</sup> is better than “ml” on RW and semantic task, while performing worse on syntactic task.

Figure 4 shows the **influence of dimensionality** *d* for SVD, 1TON and 1TON<sup>+</sup>. Peak performance for different data sets and methods is reached for  $d \in [100, 500]$ . There are no big differences in the averages across data sets and methods for high enough *d*, roughly in the interval  $[150, 500]$ . In summary, as long as *d* is chosen to be large enough (e.g.,  $\geq 150$ ), performance is robust.

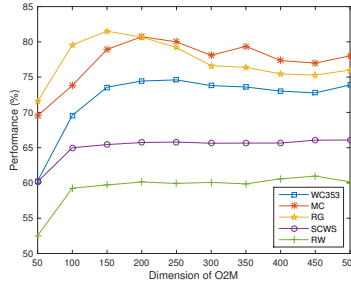
We will release the meta-embeddings produced by methods SVD, 1TON and 1TON<sup>+</sup> for  $d = 200$  and also the meta-embeddings for method CONC.

## 6.2 Domain Adaptation for POS Tagging

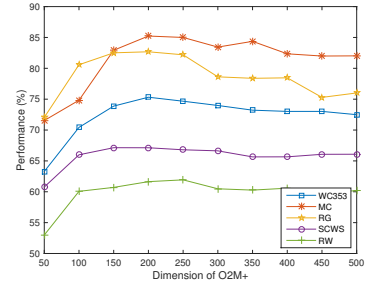
This section evaluates individual embedding sets and meta-embeddings on POS tagging. FLORS (Schnabel and Schütze, 2014), the best performing POS tagger for unsupervised domain adaptation, acts as testbed, in which POS tagging is a window-based, multilabel classification problem using a linear SVM. A word’s representation consists of *four* feature vectors based on suffix, shape, left and right distributional neighbors respectively. We



(a) Performance vs.  $d$  of SVD



(b) Performance vs.  $d$  of 1TON



(c) Performance vs.  $d$  of 1TON<sup>+</sup>

Figure 4: Influence of dimensionality

		newsgroups		reviews		weblogs		answers		emails		wsj	
		ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
baselines	TnT	88.66	54.73	90.40	56.75	93.33	74.17	88.55	48.32	88.14	58.09	95.76	88.30
	Stanford	89.11	56.02	91.43	58.66	94.15	77.13	88.92	49.30	88.68	58.42	96.83	90.25
	SVMTool	89.14	53.82	91.30	54.20	94.21	76.44	88.96	47.25	88.64	56.37	96.63	87.96
	C&P	89.51	57.23	91.58	59.67	94.41	78.46	89.08	48.46	88.74	58.62	96.78	88.65
	FLORS	90.86	66.42	92.95	75.29	94.71	83.64	90.30	62.15	89.44	62.61	96.59	90.37
+indiv	FLORS+HLBL	90.01	62.64	92.54	74.19	94.19	79.55	90.25	62.06	89.33	62.32	96.53	91.03
	FLORS+Huang	90.68	68.53	92.86	77.88	94.71	84.66	90.62	65.04	89.62	64.46	96.65	91.69
	FLORS+GloVe	90.99	70.64	92.84	78.19	94.69	86.16	90.54	65.16	89.75	65.61	96.65	92.03
	FLORS+CW	90.37	69.31	92.56	77.65	94.62	84.82	90.23	64.97	89.32	65.75	96.58	91.36
	FLORS+W2V	90.72	72.74	92.50	77.65	94.75	86.69	90.26	64.91	89.19	63.75	96.40	91.03
+meta	FLORS+CONC	<b>91.87</b>	72.64	<b>92.92</b>	<b>78.34</b>	<b>95.37</b>	86.69	<b>90.69</b>	<b>65.77</b>	<b>89.94</b>	<b>66.90</b>	<b>97.31</b>	<b>92.69</b>
	FLORS+SVD	90.98	70.94	92.47	77.88	94.50	86.49	<b>90.75</b>	64.85	<b>89.88</b>	<b>65.99</b>	96.42	90.36
	FLORS+1TON	<b>91.53</b>	<b>72.84</b>	<b>93.58</b>	78.19	<b>95.65</b>	<b>87.62</b>	<b>91.36</b>	<b>65.36</b>	<b>90.31</b>	<b>66.48</b>	<b>97.66</b>	<b>92.86</b>
	FLORS+1TON <sup>+</sup>	<b>91.86</b>	<b>73.36</b>	<b>93.14</b>	<b>78.77</b>	<b>95.65</b>	<b>87.29</b>	<b>91.73</b>	<b>66.28</b>	<b>90.53</b>	<b>66.72</b>	<b>97.75</b>	<b>92.55</b>

Table 4: POS tagging results on six target domains. “baselines” lists representative systems for this task, including FLORS. “+indiv / +meta”: FLORS with individual embedding set / meta-embeddings. Bold means higher than “baselines” and “+indiv”.

insert word’s embedding as the *fifth* feature vector. All embedding sets (except for 1TON<sup>+</sup>) are extended to the union vocabulary by MUTUALLEARNING. We follow Schnabel and Schütze (2014) for all feature learning and also train on sections 2-21 of Wall Street Journal (WSJ) and evaluate on the development sets of six different target domains: five SANCL (Petrov and McDonald, 2012) domains – newsgroups, weblogs, reviews, answers, emails – and sections 22-23 of WSJ for in-domain testing.

Table 4 gives results for some representative systems (“baselines”), FLORS with individual embedding sets (“+indiv”) and FLORS with meta-embeddings (“+meta”). Following conclusions can be drawn. (i) Not all individual embedding sets are beneficial in this task; e.g., HLBL embeddings make FLORS perform worse in 11 out of 12 cases. (ii) However, in most cases, embeddings improve

system performance, which is consistent with prior work on using embeddings for this type of task (Xiao and Guo, 2013; Yang and Eisenstein, 2014; Tsuboi, 2014). (iii) Meta-embeddings generally help more than the individual embedding sets, except for SVD (which only performs better in 3 out of 12 cases).

## 7 Conclusion

This work presented four ensemble methods – CONC, SVD, 1TON and 1TON<sup>+</sup> – for learning meta-embeddings from multiple embedding sets. Experiments on word similarity, word analogy and POS tagging indicated the high quality of these meta-embeddings. The ensemble methods have the added advantage of increasing vocabulary coverage. We will release the meta-embeddings.



## References

- Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.
- Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.
- Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.
- Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Not all neural embeddings are born equal. In *NIPS Workshop on Learning Semantics*.
- Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2015a. Embedding word similarity with neural machine translation. *ICLR workshop*.
- Felix Hill, Roi Reichart, and Anna Korhonen. 2015b. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.
- Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.
- Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on Machine learning*.
- Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.
- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- Tomáš Mikolov. 2012. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*.
- George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.
- Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12.
- Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.
- Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.
- Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.
- Tobias Schnabel and Hinrich Schütze. 2014. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26.
- Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

- Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 938–950.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.
- Min Xiao and Yuhong Guo. 2013. Domain adaptation for sequence labeling tasks with a probabilistic language adaptation model. In *Proceedings of The 30th International Conference on Machine Learning*, pages 293–301.
- Yi Yang and Jacob Eisenstein. 2014. Unsupervised domain adaptation with feature embeddings. *ICLR workshop*.