A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS

Sanjeev Arora, Yingyu Liang, Tengyu Ma
Princeton University
{arora,yingyul,tengyu}@cs.princeton.edu

ABSTRACT

The success of neural network methods for computing word embeddings has motivated methods for generating semantic embeddings of longer pieces of text, such as sentences and paragraphs. Surprisingly, Wieting et al (ICLR'16) showed that such complicated methods are outperformed, especially in out-of-domain (transfer learning) settings, by simpler methods involving mild retraining of word embeddings and basic linear regression. The method of Wieting et al. requires retraining with a substantial labeled dataset such as Paraphrase Database (Ganitkevitch et al., 2013).

The current paper goes further, showing that the following completely unsupervised sentence embedding is a formidable baseline: Use word embeddings computed using one of the popular methods on unlabeled corpus like Wikipedia, represent the sentence by a weighted average of the word vectors, and then modify them a bit using PCA/SVD. This weighting improves performance by about 10% to 30% in textual similarity tasks, and beats sophisticated supervised methods including RNN's and LSTM's. It even improves Wieting et al.'s embeddings. This simple method should be used as the baseline to beat in future, especially when labeled training data is scarce or nonexistent.

The paper also gives a theoretical explanation of the success of the above unsupervised method using a latent variable generative model for sentences, which is a simple extension of the model in Arora et al. (TACL'16) with new "smoothing" terms that allow for words occurring out of context, as well as high probabilities for words like *and*, *not* in all contexts.

1 INTRODUCTION

Word embeddings computed using diverse methods are basic building blocks for Natural Language Processing (NLP) and Information Retrieval (IR). They capturing the similarities between words (e.g., (Bengio et al., 2003; Collobert & Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014)). Recent work has tried to compute embeddings that capture the semantics of word sequences (phrases, sentences, and paragraphs), with methods ranging from simple additional composition of the word vectors to sophisticated architectures such as convolutional neural networks and recurrent neural networks (e.g., (Iyyer et al., 2015; Le & Mikolov, 2014; Kiros et al., 2015; Socher et al., 2011; Blunsom et al., 2014; Tai et al., 2015; Wang et al., 2016)). Recently, (Wieting et al., 2016) learned general-purpose, paraphrastic sentence embeddings by starting with standard word embeddings and modifying them based on supervision from the Paraphrase pairs dataset (PPDB), and constructing sentence embeddings by training a simple word averaging model. This simple method leads to better performance on textual similarity tasks than a wide variety of methods and serves as a good initialization for textual classification tasks. However, supervision from the paraphrase dataset seems crucial, since they report that simple average of the initial word embeddings does not work very well.

Here we give a new sentence embedding method that is embarrassingly simple: just compute the weighted average of the word vectors in the sentence and then remove the projections of the average vectors on their first principal component ("*common component removal*"). Here the weight of a word w is $a/(a + p(w))$ with a being a parameter and $p(w)$ the (estimated) word frequency; we call this *smooth inverse frequency* (SIF). This method achieves significantly better performance than the unweighted average on a variety of textual similarity tasks, and on most of these tasks even beats some sophisticated supervised methods tested in (Wieting et al., 2016), including some RNN and LSTM models. The method is well-suited for domain adaptation settings, i.e., word vectors trained on various kinds of corpora are used for computing the sentence embeddings in different testbeds. It is also fairly robust to the weighting scheme: using the word frequencies estimated from different corpora does not harm the performance; a wide range of the parameters a can achieve close-to-best results, and an even wider range can achieve significant improvement over unweighted average.

Of course, this SIF reweighting is highly reminiscent of TF-IDF reweighting from information retrieval (Sparck Jones, 1972; Robertson, 2004) if one treats a "sentence" as a "document" and make the reasonable assumption that the sentence doesn't typically contain repeated words. Such reweightings (or related ideas like removing frequent words from the vocabulary) are a good rule of thumb but has not had theoretical justification in a word embedding setting.

The current paper provides a theoretical justification for the reweighting using a generative model for sentences, which is a simple modification for the Random Walk on Discourses model for generating text in (Arora et al., 2016). In that paper, it was noted that the model theoretically implies a sentence embedding, namely, simple average of embeddings of all the words in it.

We modify this theoretical model, motivated by the empirical observation that most word embedding methods, since they seek to capture word cooccurence probabilities using vector inner product, end up giving large vectors to frequent words, as well as giving unnecessarily large inner products to word pairs, simply to fit the empirical observation that words sometimes occur out of context in documents. These anomalies cause the average of word vectors to have huge components along semantically meaningless directions. Our modification to the generative model of (Arora et al., 2016) allows "smoothing" terms, and then a max likelihood calculation leads to our SIF reweighting.

Interestingly, this theoretically derived SIF does better (by a few percent points) than traditional TF-IDF in our setting. The method also improves the sentence embeddings of Wieting et al., as seen in Table 1. Finally, we discovered that —contrary to widespread belief—**Word2Vec**(CBOW) also does *not* use simple average of word vectors in the model, as misleadingly suggested by the usual expression $\Pr[w|w_1, w_2, \ldots, w_5] \propto \exp(v_w \cdot (\frac{1}{5}\sum_i v_{w_i}))$. A dig into the implementation shows it implicitly uses a weighted average of word vectors —again, different from TF-IDF— and this weighting turns out to be quite similar in effect to ours. (See Section 3.1.)

2 Related Work

Word embeddings. Word embedding methods represent words as continuous vectors in a low dimensional space which capture lexical and semantic properties of words. They can be obtained from the internal representations from neural network models of text (Bengio et al., 2003; Collobert & Weston, 2008; Mikolov et al., 2013a) or by low rank approximation of co-occurrence statistics (Deerwester et al., 1990; Pennington et al., 2014). The two approaches are known to be closely related (Levy & Goldberg, 2014; Hashimoto et al., 2016; Arora et al., 2016).

Our work is most directly related work to (Arora et al., 2016), which proposed a random walk model for generating words in the documents. Our sentence vector can be seen as approximate inference of the latent variables in their generative model.

Phrase/Sentence/Paragraph embeddings. Previous works have computed phrase or sentence embeddings by composing word embeddings using operations on vectors and matrices e.g., (Mitchell & Lapata, 2008; 2010; Blacoe & Lapata, 2012). They found that coordinate-wise multiplication of the vectors performed very well among the binary operations studied. Unweighted averaging is also found to do well in representing short phrases (Mikolov et al., 2013a). Another approach is recursive neural networks (RNNs) defined on the parse tree, trained with supervision (Socher et al., 2011) or without (Socher et al., 2014). Simple RNNs can be viewed as a special case where the parse tree is replaced by a simple linear chain. For example, the skip-gram model (Mikolov et al., 2013b) is extended to incorporate a latent vector for the sequence, or to treat the sequences rather than the word as basic units. In (Le & Mikolov, 2014) each paragraph was assumed to have a latent paragraph vector, which influences the distribution of the words in the paragraph. **Skip-thought** of (Kiros et al., 2015) tries to reconstruct the surrounding sentences from surrounded one and treats the hidden parameters as their vector representations. RNNs using long short-term memory (LSTM) capture long-distance dependency and have also been used for modeling sentences (Tai et al., 2015). Other neural network structures include convolution neural networks, such as (Blunsom et al., 2014) that uses a dynamic pooling to handle input sentences of varying length and do well in sentiment prediction and classification tasks.

The directed inspiration for our work is (Wieting et al., 2016) which learned paraphrastic sentence embeddings by using simple word averaging and also updating standard word embeddings based on supervision from paraphrase pairs; the supervision being used for both initialization and training.

3 A Simple Method for Sentence Embedding

We briefly recall the latent variable generative model for text in (Arora et al., 2016). The model treats corpus generation as a dynamic process, where the t-th word is produced at step t. The process is driven by the random walk of a discourse vector $c_t \in \Re^d$. Each word w in the vocabulary has a vector in $\Re^d$ as well; these are latent variables of the model. The discourse vector represents "what is being talked about." The inner product between the discourse vector c_t and the (time-invariant) word vector v_w for word w captures the correlations between the discourse and the word. The probability of observing a word w at time t is given by a log-linear word production model from Mnih and Hinton:

$$\Pr[w \text{ emitted at time } t \mid c_t] \propto \exp\left(\langle c_t, v_w \rangle\right). \tag{1}$$

The discourse vector c_t does a slow random walk (meaning that c_{t+1} is obtained from c_t by adding a small random displacement vector), so that nearby words are generated under similar discourses. It was shown in (Arora et al., 2016) that under some reasonable assumptions this model generates behavior –in terms of word-word cooccurence probabilities—that fits empirical works like word2vec and Glove. The random walk model can be relaxed to allow occasional big jumps in c_t, since a simple calculation shows that they have negligible effect on cooccurrence probabilities of words. The word vectors computed using this model are reported to be similar to those from Glove and word2vec(CBOW).

Our improved Random Walk model. Clearly, it is tempting to define the sentence embedding as follows: given a sentence s, do a MAP estimate of the discourse vectors that govern this sentence. We note that we assume the discourse vector c_t doesn't change much while the words in the sentence were emitted, and thus we can replace for simplicity all the c_t's in the sentence s by a single discourse vector c_s. In the paper (Arora et al., 2016), it was shown that the MAP estimate of c_s is —up to multiplication by scalar—the average of the embeddings of the words in the sentence.

In this paper, towards more realistic modeling, we change the model (1) as follows. This model has two types of "smoothing term", which are meant to account for the fact that some words occur out of context, and that some frequent words (presumably "the", "and " etc.) appear often regardless of the discourse. We first introduce an additive term $\alpha p(w)$ in the log-linear model, where $p(w)$ is the unigram probability (in the entire corpus) of word and α is a scalar. This allows words to occur even if their vectors have very low inner products with c_s. Secondly, we introduce a common discourse vector $c_0 \in \Re^d$ which serves as a correction term for the most frequent discourse that is often related to syntax. (Other possible correction is left to future work.) It boosts the co-occurrence probability of words that have a high component along c_0.

Concretely, given the discourse vector c_s, the probability of a word w is emitted in the sentence s is modeled by,

$$\Pr[w \text{ emitted in sentence } s \mid c_s] = \alpha p(w) + (1-\alpha)\frac{\exp\left(\langle \tilde{c}_s, v_w \rangle\right)}{Z_{\tilde{c}_s}}, \tag{2}$$

$$\text{where } \tilde{c}_s = \beta c_0 + (1-\beta)c_s, \;\; c_0 \perp c_s$$

where α and β are scalar hyperparameters, and $Z_{\tilde{c}_s} = \sum_{w\in\mathcal{V}} \exp\left(\langle \tilde{c}_s, v_w \rangle\right)$ is the normalizing constant (the partition function). We see that the model allows a word w unrelated to the discourse c_s to be omitted for two reasons: a) by chance from the term $\alpha p(w)$; b) if w is correlated with the common discourse vector c_0.

Algorithm 1 Sentence Embedding

```
Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter $a$ and estimated marginal
    probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
for all sentence $s$ in $\mathcal{S}$ do
    $v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
end for
Compute the first principal component $u$ of $\{v_s : s \in \mathcal{S}\}$
for all sentence $s$ in $\mathcal{S}$ do
    $v_s \leftarrow v_s - u u^\top v_s$
end for
```

Computing the sentence embedding. The word embeddings yielded by our model are actually the same (up to rescaling) as those by model (1). (In fact the new model was discovered by our detecting the common component c_0 in existing embeddings.) The sentence embedding will be defined as the max likelihood estimate for the vector c_s that generated it. (In this case MLE is the same as MAP since the prior is uniform.) We borrow the key modeling assumption of (Arora et al., 2016), namely that the word v_w's are roughly uniformly dispersed, which implies that the partition function Z_c is roughly the same in all directions. So assume that $Z_{\tilde{c}_s}$ is roughly the same, say Z for all $\tilde{c}_s$. By the model (2) the likelihood for the sentence is

$$p[s \mid c_s] = \prod_{w \in s} p(w \mid c_s) = \prod_{w \in s} \left[\alpha p(w) + (1-\alpha) \frac{\exp\left(\langle v_w, \tilde{c}_s \rangle\right)}{Z} \right].$$

Let

$$f_w(\tilde{c}_s) = \log \left[\alpha p(w) + (1-\alpha) \frac{\exp\left(\langle v_w, \tilde{c}_s \rangle\right)}{Z} \right]$$

denote the log likelihood of sentence s. Then, by simple calculus we have,

$$\nabla f_w(\tilde{c}_s) = \frac{1}{\alpha p(w) + (1-\alpha) \exp\left(\langle v_w, \tilde{c}_s \rangle\right)/Z} \frac{1-\alpha}{Z} \exp\left(\langle v_w, \tilde{c}_s \rangle\right) v_w.$$

Then by Taylor expansion, we have,

$$\begin{aligned} f_w(\tilde{c}_s) &\approx f_w(0) + \nabla f_w(0)^\top \tilde{c}_s \\ &= \text{constant} + \frac{(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)} \left\langle v_w, \tilde{c}_s \right\rangle. \end{aligned}$$

Therefore, the maximum likelihood estimator for $\tilde{c}_s$ on the unit sphere (ignoring normalization) is approximately,[1]

$$\arg\max \sum_{w \in s} f_w(\tilde{c}_s) \propto \sum_{w \in s} \frac{a}{p(w) + a} v_w, \text{ where } a = \frac{1-\alpha}{\alpha Z}. \tag{3}$$

That is, the MLE is approximately a weighted average of the vectors of the words in the sentence. Note that for more frequent words w, the weight $a/(p(w)+a)$ is smaller, so this naturally leads to a down weighting of the frequent words.

To estimate c_s, we estimate the direction c_0 by computing the first principal component of $\tilde{c}_s$'s for a set of sentences. In other words, the final sentence embedding is obtained by subtracting the projection of $\tilde{c}_s$'s to their first principal component. This is summarized in Algorithm 1.

3.1 Connection to Subsampling Probabilities in word2vec

Word2vec (Mikolov et al., 2013b) uses a sub-sampling technique which downsamples word w with probability proportional to $1/\sqrt{p(w)}$ where $p(w)$ is the marginal probability of the word w. This

[1] Note that $\max_{c:\|c\|=1} C + \langle c, g \rangle = g/\|g\|$ for any constant C.

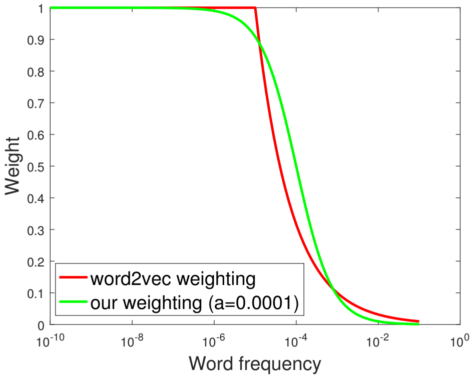


Figure 1: The subsampling probabilities in word2vec are similar to our weighting scheme.

heuristic not only speeds up the training but also learns more regular word representations. Here we explain that this corresponds to an implicit reweighting of the word vectors in the model and therefore the statistical benefit should be of no surprise.

Recall the vanilla CBOW model of word2vec:

$$\Pr[w_t \mid w_{t-1}, \dots, w_{t-5}] \propto \exp\left(\langle \bar{v}_t, v_w \rangle\right), \text{ where } \bar{v}_t = \frac{1}{5}\sum_{i=1}^{5} v_{w_{t-i}}. \tag{4}$$

It can be shown that the loss (MLE) for the single word vector v_w (from this occurrence) can be abstractly written in the form,

$$g(v_w) = \gamma(\langle \bar{v}_t, v_w \rangle) + \text{negative sampling terms},$$

where $\gamma(x) = \log(1/(1+e^{-x}))$ is the logistic function. Therefore, the gradient of $g(v_w)$ is

$$\nabla g(v_w) = \gamma'(\langle \bar{v}_t, v_w \rangle)\bar{v}_t = \alpha(v_{w_{t-5}} + v_{w_{t-4}} + v_{w_{t-3}} + v_{w_{t-2}} + v_{w_{t-1}}), \tag{5}$$

where α is a scalar. That is, *without* the sub-sampling trick, the update direction is the average of the word vectors in the window.

The sub-sampling trick in (Mikolov et al., 2013b) randomly selects the summands in equation (5) to “estimate” the gradient. Specifically, the sampled update direction is

$$\tilde{\nabla} g(v_w) = \alpha(J_5 v_{w_{t-5}} + J_4 v_{w_{t-4}} + J_3 v_{w_{t-3}} + J_2 v_{w_{t-2}} + J_1 v_{w_{t-1}}) \tag{6}$$

where J_k's are Bernoulli random variables with $\Pr[J_k = 1] = q(w_{t-k}) \triangleq \min\left\{1, \sqrt{\frac{10^{-5}}{p(w_{t-k})}}\right\}$. However, we note that $\tilde{\nabla} g(v_w)$ is (very) *biased* estimator! We have that the expectation of $\tilde{\nabla} g(v_w)$ is a *weighted* sum of the word vectors,

$$\mathbb{E}\left[\tilde{\nabla} g(v_w)\right] = \alpha(q(w_{t-5})v_{w_{t-5}} + q(w_{t-4})v_{w_{t-4}} + q(w_{t-3})v_{w_{t-3}} + q(w_{t-2})v_{w_{t-2}} + q(w_{t-1})v_{w_{t-1}}).$$

In fact, the expectation $\mathbb{E}[\tilde{\nabla} g(v_w)]$ corresponds to the gradient of a modified word2vec model with the average $\bar{v}_t$ (in equation (4)) being replaced by the weighted average $\sum_{k=1}^{5} q(w_{t-k})v_{w_{t-k}}$. Such a weighted model can also share the same form of what we derive form our random walk model as in equation (3). Moreover, the weighting $q(w_i)$ closely tracks our weighting scheme $a/(a + p(w))$ when using parameter $a = 10^{-4}$; see Figure 1 for an illustration. Therefore, the expected gradient here is approximately the estimated discourse vector in our model! Thus, word2vec with sub-sampling gradient heuristic corresponds to a stochastic gradient update method for using our weighting scheme.

	Results collected from (Wieting et al., 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

Table 1: Experimental results (Pearson's $r \times 100$) on textual similarity tasks. The highest score in each row is in boldface. The methods can be supervised (denoted as Su.), semi-supervised (Se.), or unsupervised (Un.). See the main text for the description of the methods.

4 Experiments

4.1 Textual Similarity Tasks

Datasets. We test our methods on the 22 textual similarity datasets including all the datasets from SemEval semantic textual similarity (STS) tasks (2012-2015) (Agirre et al., 2012; 2013; 2014; Agirrea et al., 2015), and the SemEval 2015 Twitter task (Xu et al., 2015) and the SemEval 2014 Semantic Relatedness task (Marelli et al., 2014). The objective of these tasks is to predict the similarity between two given sentences. The evaluation criterion is the Pearson's coefficient between the predicted scores and the ground-truth scores.

Experimental settings. We will compare our method with the following:

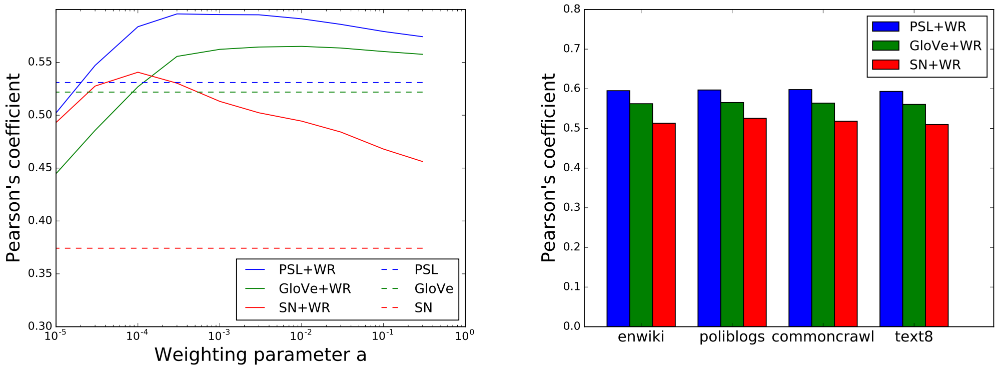
1. Unsupervised: ST, avg-GloVe, tfidf-GloVe. ST denotes the skip-thought vectors (Kiros et al., 2015), avg-GloVe denotes the unweighted average of the GloVe vectors (Pennington et al., 2014),[2] and tfidf-GloVe denotes the weighted average of GloVe vectors using TF-IDF weights.
2. Semi-supervised: avg-PSL. This method uses the unweighted average of the PARAGRAM-SL999 (PSL) word vectors from (Wieting et al., 2015). The word vectors are trained using labeled data, but the sentence embedding are computed by unweighted average without training.
3. Supervised: PP, PP-proj., DAN, RNN, iRNN, LSTM (o.g.), LSTM (no). All these methods are initialized with PSL word vectors and then trained on the PPDB dataset. PP and PP-proj. are proposed in (Wieting et al., 2016). The first is an average of the word vectors, and the second additionally adds a linear projection. The word vectors are updated during the training. DAN denotes the deep averaging network of (Iyyer et al., 2015). RNN denotes the classical recurrent neural network, and iRNN denotes a variant with the activation being the identity, and the weight matrices initialized to identity. The LSTM is the version from (Gers et al., 2002), either with output gates (denoted as LSTM(o.g.)) or without (denoted as LSTM (no)).

Our method can be applied to any types of word embeddings. To get a completely unsupervised method, we apply it to the GloVe vectors. The weighting parameter a is fixed to 10^{-3}, and the word frequencies $p(w)$ are estimated from the commoncrawl dataset.[3] This is denoted by GloVe+WR in Table 1. We also apply our method on the PSL vectors, denoted as PSL+WR, which is a semi-supervised method.

Results. The results are reported in Table 1. Each year there are 4 to 6 STS tasks. For clarity, we only report the average result for the STS tasks each year; the detailed results are in the appendix.

[2] We used the vectors that are publicly available at http://nlp.stanford.edu/projects/glove/. They are 300-dimensional vectors that were trained on the 840 billion token Common Crawl corpus.

[3] It is possible to tune the parameter a to get better results. The effect of a and the corpus for estimating word frequencies are studied in Section 4.1.1.



(a) (b)

Figure 2: Effect of weighting scheme in our method on the average performance on STS 2012 tasks. Best viewed in color. (a) Performance v.s. weighting parameter a. Three types of word vectors (PSL, GloVe, SN) are tested using $p(w)$ estimated on the enwiki dataset. The best performance is usually achieved at $a = 10^{-3}$ to $a = 10^{-4}$. (b) Performance v.s. datasets used for estimating $p(w)$. Four datasets (enwiki, poliblogs, commoncrawl, text8) are used to estimate $p(w)$ which is then used in our method. The parameter a is fixed to be 10^{-3}. The performance is almost the same for different settings.

The unsupervised method GloVe+WR improves upon avg-GloVe significantly by 10% to 30%, and beats the baselines by large margins. It achieves better performance than LSTM and RNN and comparable to DAN, even though the later three use supervision. This demonstrates the power of this simple method: it can be even stronger than highly-tuned supervisedly trained sophisticated models. Using TF-IDF weighting scheme also improves over the unweighted average, but not as much as our method.

The semi-supervised method PSL+WR achieves the best results for four out of the six tasks and are comparable to the best in the rest of two tasks. Overall, it outperforms the avg-PSL baseline and all the supervised models initialized with the same PSL vectors. This demonstrates the advantage of our method over the training for those models.

Finally, in the appendix, we showed that our two ideas all contribute to the improvement: for GloVe vectors, using smooth inverse frequency weighting alone improves over unweighted average by about 5%, using common component removal alone improves by 10%, and using both improves by 13%.

4.1.1 Effect of Weighting Parameter on Performance

We study the sensitivity of our method to the weighting parameter a, the method for computing word vectors, and the estimated word probabilities $p(w)$. First, we test the performance of three types of word vectors (PSL, GloVe, and SN) on the STS 2012 tasks. SN vectors are trained on the enwiki dataset (Wikimedia, 2012) using the method in (Arora et al., 2016), while PSL and GloVe vectors are those used in Table 1. We enumerate $a \in \{10^{-i}, 3 \times 10^{-i} : 1 \leq i \leq 5\}$ and use the $p(w)$ estimated on the enwiki dataset. Figure 2a shows that for all three kinds of word vectors, a wide range of a leads to significantly improved performance over the unweighted average. Best performance occurs from $a = 10^{-3}$ to $a = 10^{-4}$.

Next, we fix $a = 10^{-3}$ and use four very different datasets to estimate $p(w)$: enwiki (wikipedia, 3 billion tokens), poliblogs (Yano et al., 2009) (political blogs, 5 million), commoncrawl (Buck et al., 2014) (Internet crawl, 800 billion), text8 (Mahoney, 2008) (wiki subset, 1 million). Figure 2b shows performance is almost the same for all four settings.

	PP	DAN	RNN	LSTM (no)	LSTM (o.g.)	skip-thought	Ours
similarity (SICK)	84.9	85.96	73.13	85.45	83.41	85.8	**86.03**
entailment (SICK)	83.1	84.5	76.4	83.2	82.0	-	**84.6**
sentiment (SST)	79.4	83.4	86.5	86.6	**89.2**	-	82.2

Table 2: Results on similarity, entailment, and sentiment tasks. The sentence embeddings are computed unsupervisedly, and then used as features in downstream supervised tasks. The row for similarity (SICK) shows Pearson's $r \times 100$ and the last two rows show accuracy. The highest score in each row is in boldface. Results in Column 2 to 6 are collected from (Wieting et al., 2016), and those in Column 7 for skip-thought are from (Lei Ba et al., 2016).

The fact that our method can be applied on different types of word vectors trained on different corpora also suggests it should be useful across different domains. This is especially important for unsupervised methods, since the unlabeled data available may be collected in a different domain from the target application.

4.2 Supervised tasks

The sentence embeddings obtained by our method can be used as features for downstream supervised tasks. We consider three tasks: the SICK similarity task, the SICK entailment task, and the Stanford Sentiment Treebank (SST) binary classification task (Socher et al., 2013). To highlight the representation power of the sentence embeddings learned unsupervisedly, we fix the embeddings and only learn the classifier. Setup of supervised tasks mostly follow (Wieting et al., 2016) to allow fair comparison, i.e., the classifier a linear projection followed by the classifier in (Kiros et al., 2015). The linear projection maps the sentence embeddings into 2400 dimension (the same as the skip-thought vectors), and is learned during the training. We compare our method to PP, DAN, RNN, and LSTM, which are the methods used in Section 4.1. We also compare to the skip-thought vectors (with improved training in (Lei Ba et al., 2016)).

Results. Our method gets better or comparable performance compared to the competitors. It gets the best results for two of the tasks. This demonstrates the power of our simple method. We emphasize that our embeddings are unsupervisedly learned, while DAN, RNN, LSTM are trained with supervision. Furthermore, skip-thought vectors are much higher dimensional than ours (though projected into higher dimension, the original 300 dimensional embeddings contain all the information).

The advantage is not as significant as in the textual similarity tasks. This is possibly because similarity tasks rely directly upon cosine similarity, which favors our method's approach of removing the common components (which can be viewed as a form of denoising), while in supervised tasks, with the cost of some label information, the classifier can pick out the useful components and ignore the common ones.

5 Conclusions

This work provided a simple approach to sentence embedding, based on the discourse vectors in the random walk model for generating text (Arora et al., 2016). It is simple and unsupervised, but achieves significantly better performance than baselines on various textual similarity tasks, and can even beat sophisticated supervised methods such as some RNN and LSTM models. The sentence embeddings obtained can be used as features in downstream supervised tasks, which also leads to better or comparable results compared to the sophisticated methods.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 385–393. Association for Computational Linguistics, 2012.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. Sem 2013 shared task: Semantic textual similarity. in second joint conference on lexical and computational semantics. In *Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, 2013.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 81–91, 2014.

Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalara, Rada Mihalceab, et al. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pp. 252–263, 2015.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. *Transaction of Association for Computational Linguistics*, 2016.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003.

William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, 2014.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 2002.

Tatsunori B. Hashimoto, David Alvarez-Melis, and Tommi S. Jaakkola. Word embeddings as metric recovery in semantic spaces. *Transactions of the Association for Computational Linguistics*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*, 2015.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, 2015.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, 2016.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, 2014.

Matt Mahoney. Wikipedia text preprocess script. http://mattmahoney.net/dc/textdata.html, 2008. Accessed Mar-2015.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013b.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Association for Computational Linguistics*, 2008.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 2010.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme2, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing*, 2014.

Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 2004.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 2011.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, 2013.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2014.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. Cse: Conceptual sentence embeddings based on attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 2015.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations*, 2016.

Wikimedia. English Wikipedia dump. http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, 2012. Accessed Mar-2015.

Wei Xu, Chris Callison-Burch, and William B Dolan. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of SemEval*, 2015.

Tae Yano, William W Cohen, and Noah A Smith. Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

A Details of Experimental Setting

A.1 Unsupervised Task: Textual Similarity

The competitors. We give a brief overview of the competitors. RNN is the classical recurrent neural network:

$$h_t = f(W_x W_w^{x_t} + W_h h_{t-1} + b)$$

where f is the activation, W_x, W_h and b are parameters, and x_t is the t-th token in the sentence. The sentence embedding of RNN is just the hidden vector of the last token. iRNN is a special RNN with the activation being the identity, the weight matrices initialized to identity, and b initialized to zero. LSTM (Hochreiter & Schmidhuber, 1997) is a recurrent neural network architecture designed to capture long-distance dependencies. Here, the version from (Gers et al., 2002) is used.

The supervised methods are initialized with PARAGRAM-SL999 (PSL) vectors, and trained using the approach of (Wieting et al., 2016) on the XL section of the PPDB data (Pavlick et al., 2015) which contains about 3 million unique phrase pairs. After training, the final models can be used to generate sentence embeddings on the test data. For hyperparameter tuning they used 100k examples sampled from PPDB XXL and trained for 5 epochs. Then after finding the hyperparameters that maximize Spearmans coefficients on the Pavlick et al. PPDB task, they are trained on the entire XL section of PPDB for 10 epochs. See (Wieting et al., 2016) and related papers for more details about these methods.

The tfidf-GloVe method is a weighted average of the GloVe vectors, where the weights are defined by the TF-IDF scheme. More precisely, the embedding of a sentence s is

$$v_s = \frac{1}{|s|} \sum_{w \in s} \mathrm{IDF}_w v_w \tag{7}$$

where IDF_w is the inverse document frequency of w, and $|s|$ denotes the number of words in the sentence. Here, the TF part of the TF-IDF scheme is taken into account by the sum over $w \in s$. Furthermore, when computing IDF_w, each sentence is viewed as a "document":

$$\mathrm{IDF}_w := \log \frac{1+N}{1+N_w}$$

where N is the total number of sentences and N_w is the number of sentences containing w, and 1 is added to avoid division by 0. In the experiments, we use all the textual similarity datasets to compute IDF_w.

Detailed experimental results. In the main body we present the average results for STS tasks by year. Each year there are actually 4 to 6 STS tasks, as shown in Table 3. Note that tasks with the same name in different years are actually different tasks. Here we provide the results for each tasks in Table 4. PSL+WR achieves the best results on 12 out of 22 tasks, PP and PP-proj. achieves on 3, tfidf-GloVe achieves on 2, and DAN, iRNN, and GloVe+WR achieves on 1. In general, our method improves the performance significantly compared to the unweighted average, though on rare cases such as MSRpar it can decrease the performance.

STS'12	STS'13	STS'14	STS'15
MSRpar	headline	deft forum	anwsers-forums
MSRvid	OnWN	deft news	answers-students
SMT-eur	FNWN	headline	belief
OnWN	SMT	images	headline
SMT-news		OnWN	images
		tweet news	

Table 3: The STS tasks by year. Note that tasks with the same name in different years are actually different tasks.

	Results collected from (Wieting et al., 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
MSRpar	42.6	43.7	40.3	18.6	43.4	16.1	9.3	16.8	47.7	**50.3**	41.6	35.6	43.3
MSRvid	74.5	74.0	70.0	66.5	73.4	71.3	71.3	41.7	63.9	77.9	60.0	83.8	**84.1**
SMT-eur	47.3	49.4	43.8	40.9	47.1	41.8	44.3	35.2	46.0	**54.7**	42.4	49.9	44.8
OnWN	70.6	70.1	65.9	63.1	70.1	65.2	56.4	29.7	55.1	64.7	63.0	66.2	**71.8**
SMT-news	58.4	**62.8**	60.0	51.3	58.1	60.8	51.0	30.8	49.6	45.7	57.0	45.6	53.6
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
headline	72.4	72.6	71.2	59.5	72.8	57.4	48.5	34.6	63.8	69.2	68.8	69.2	**74.1**
OnWN	67.7	68.0	64.1	54.6	69.4	68.5	50.4	10.0	49.0	72.9	48.0	**82.8**	82.0
FNWN	43.9	46.8	43.1	30.9	45.3	24.7	38.4	30.4	34.2	36.6	37.9	39.4	**52.4**
SMT	39.2	**39.8**	38.3	33.8	39.4	30.1	28.8	24.3	22.3	29.6	31.0	37.9	38.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
deft forum	48.7	51.1	49.0	41.5	49.0	44.2	46.1	12.9	27.1	37.5	37.2	41.2	**51.4**
deft news	**73.1**	72.2	71.7	53.7	72.4	52.8	39.1	23.5	68.0	68.7	67.0	69.4	72.6
headline	69.7	**70.8**	69.2	57.5	70.2	57.5	50.9	37.8	59.5	63.7	65.3	64.7	70.1
images	78.5	78.1	76.9	67.6	78.2	68.5	62.9	51.2	61.0	72.5	62.0	82.6	**84.8**
OnWN	78.8	79.5	75.7	67.7	78.8	76.9	61.7	23.3	58.4	75.2	61.1	82.8	**84.5**
tweet news	76.4	75.8	74.2	58.0	76.9	58.7	48.2	39.9	51.2	65.1	64.7	70.1	**77.5**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
answers-forum	68.3	65.1	62.6	32.8	67.4	51.9	50.7	36.1	30.5	45.6	38.8	63.9	**70.1**
answers-student	**78.2**	77.8	78.1	64.7	**78.2**	71.5	55.7	33.0	63.0	63.9	69.2	70.4	75.9
belief	**76.2**	75.4	72.0	51.9	75.9	61.7	52.6	24.6	40.5	49.5	53.2	71.8	75.3
headline	74.8	75.2	73.5	65.3	75.1	64.0	56.6	43.6	61.8	70.9	69.0	70.7	**75.9**
images	81.4	80.3	77.5	71.4	81.1	70.4	64.2	17.7	67.5	72.9	69.9	81.5	**84.1**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

Table 4: Experimental results (Pearson's $r \times 100$) on textual similarity tasks. The highest score in each row is in boldface. The methods can be supervised (denoted as Su.), semi-supervised (Se.), or unsupervised (Un.). See the main text for the description of the methods.

Effects of smooth inverse frequency and common component removal. There are two key ideas in our methods: smooth inverse frequency weighting (W) and common component removal (R). It is instructive to see their effects separately. Let GloVe+W denote the embeddings using only smooth inverse frequency, and GloVe+R denote that using only common component removal. Similarly define PSL+W and PSL+R. The results for these methods are shown in Table 5. When using GloVe vectors, W alone improves the performance of the unweighted average baseline by about 5%, R alone improves by 10%, W and R together improves by 13%. When using PSL vectors, W gets 10%, R gets 10%, W and R together gets 13%. In summary, both techniques are important for obtaining significant advantage over the unweighted average.

A.2 SUPERVISED TASKS

Setup of supervised tasks mostly follow (Wieting et al., 2016) to allow fair comparison: the sentence embeddings are fixed and fed into some classifier which are trained. For the SICK similarity task, given a pair of sentences with embeddings v_L and v_R, first do a linear projection:

$$h_L = W_p v_L, h_R = W_p v_R$$

where W_p is of size $300 \times d_p$ and is learned during training. $d_p = 2400$ matches the dimension of the skip-thought vectors. Then h_L and h_R are used in the objective function from (Tai et al., 2015). Almost the same approach is used for the entailment task.For the sentiment task, the classifier has a fully-connected layer with a sigmoid activation followed by a softmax layer.

Recall that our method has two steps: smooth inverse frequency weighting and common component removal. For the experiments here, we do not perform the common component removal, since it can be absorbed into the projection step. For the weighted average, the hyperparameter a is enumerated in $\{10^{-i}, 3 \times 10^{-i} : 2 \leq i \leq 3\}$. The other hyperparameters are enumerated as in (Wieting et al., 2016), and the same validation approach is used to select the final values.

	Unsupervised				Semi-supervised			
Tasks	avg-GloVe	GloVe+W	GloVe+R	GloVe+WR	avg-PSL	PSL+W	PSL+R	PSL+WR
MSRpar	47.7	43.6	36.4	35.6	41.6	40.9	42.5	43.3
MSRvid	63.9	78.7	79.4	83.8	60.0	80.4	76.4	84.1
SMT-eur	46.0	51.1	48.5	49.9	42.4	45.0	45.1	44.8
OnWN	55.1	54.3	68.3	66.2	63.0	67.8	71.0	71.8
SMT-news	49.6	42.2	45.6	45.6	57.0	56.2	50.7	53.6
STS'12	52.5	54.0	55.6	56.2	52.8	58.1	57.2	59.5
headline	63.8	63.8	68.9	69.2	68.8	72.6	72.7	74.1
OnWN	49.0	68.0	75.4	82.8	48.0	69.8	73.5	82.0
FNWN	34.2	23.0	34.9	39.4	37.9	49.3	40.7	52.4
SMT	22.3	29.5	36.4	37.9	31.0	39.2	37.3	38.5
STS'13	42.3	44.0	53.9	56.6	46.4	57.7	56.0	61.8
deft forum	27.1	29.1	39.8	41.2	37.2	45.8	45.3	51.4
deft news	68.0	68.5	66.6	69.4	67.0	75.1	67.4	72.6
headline	59.5	59.3	64.6	64.7	65.3	68.9	68.5	70.1
images	61.0	74.1	78.4	82.6	62.0	82.9	80.2	84.8
OnWN	58.4	68.0	77.6	82.8	61.1	77.6	77.7	84.5
tweet news	51.2	57.3	73.2	70.1	64.7	73.6	77.9	77.5
STS'14	54.2	59.4	66.7	68.5	59.5	70.7	69.5	73.5
answers-forum	30.5	41.4	58.4	63.9	38.8	56.0	61.0	70.1
answers-student	63.0	61.5	73.2	70.4	69.2	73.3	76.8	75.9
belief	40.5	47.7	69.5	71.8	53.2	64.3	71.3	75.3
headline	61.8	64.0	70.1	70.7	69.0	74.5	74.6	75.9
images	67.5	75.4	77.9	81.5	69.9	83.4	79.9	84.1
STS'15	52.7	58.0	69.8	71.7	60.0	70.3	72.7	76.3
SICK'14	65.9	70.5	70.6	72.2	66.4	73.1	70.3	72.9
Twitter'15	30.3	33.8	70.6	48.0	36.3	45.7	51.9	49.0

Table 5: Experimental results (Pearson's $r \times 100$) on textual similarity tasks using only smooth inverse frequency, using only common component removal, or using both.