Unsupervised Random Walk Sentence Embeddings: A Strong but Simple Baseline

Kawin Ethayarajh
Department of Computer Science, University of Toronto
kawin@cs.toronto.edu

Abstract

Using a random walk model of text generation, Arora et al. (2017) proposed a strong baseline for computing sentence embeddings: take a weighted average of word embeddings and modify with SVD. This simple method even outperforms far more complex approaches such as LSTMs on textual similarity tasks. In this paper, we first show that word vector length has a confounding effect on the probability of a sentence being generated in Arora et al.'s model. We propose a random walk model that is robust to this confound, where the probability of word generation is inversely related to the angular distance between the word and sentence embeddings. Our approach beats Arora et al.'s by up to 44.4% on textual similarity tasks and is competitive with state-of-the-art methods. Unlike Arora et al.'s method, ours requires no hyperparameter tuning, which means it can be used when there is no labelled data.

1 Introduction

Distributed representations of words, better known as word embeddings, have become fixtures of current methods in natural language processing. Word embeddings can be generated in a number of ways (Bengio et al., 2003; Collobert and Weston, 2008; Pennington et al., 2014; Mikolov et al., 2013) by capturing the semantics of a word using the contexts it appears in. Recent work has tried to extend that intuition to sequences of words, using methods ranging from a weighted average of word embeddings to convolutional, recursive, and recurrent neural networks (Le and Mikolov, 2014; Kiros et al., 2015; Luong et al., 2013; Tai et al., 2015). Still, Wieting et al. (2016b) found that these sophisticated architectures are often outperformed, particularly in transfer learning settings, by sentence embeddings generated as a simple average of tuned word embeddings.

Arora et al. (2017) provided a more powerful approach: compute the sentence embeddings as weighted averages of word embeddings, then subtract from each one the vector projection on their first principal component. The weighting scheme, *smoothed inverse frequency* (SIF), is derived from a random walk model where words in a sentence s are produced by the random walk of a latent discourse vector c_s. A word unrelated to c_s can be produced by chance or if it is part of frequent discourse such as stopwords. This approach evens outperforms more complex models such as LSTMs on textual similarity tasks. Arora et al. argued that the simplicity and effectiveness of their method make it a tough-to-beat baseline for sentence embeddings. Though they call their approach unsupervised, others have noted that it is actually 'weakly supervised', since it requires hyperparameter tuning (Cer et al., 2017).

In this paper, we first propose a class of worst-case scenarios for Arora et al.'s (2017) random walk model. Specifically, given some sentence g that is dominated by words with zero similarity, and some sentence h that is dominated by identical words, we show that their approach can return two discourse vectors c_g and c_h such that $p(g|c_g) \approx p(h|c_h)$, provided that the word vectors for g have a sufficiently greater length than those for h. In other words, word vector length has a confounding effect on the probability of a sentence being generated, and this effect can be strong enough to yield completely unintuitive results. This problem is not endemic to these scenarios, though they are the most illustrative of it; because of the underlying log-linear word production model, Arora et al.'s model is fundamentally

Proceedings of the 3rd Workshop on Representation Learning for NLP, pages 91–100
Melbourne, Australia, July 20, 2018.

sensitive to word vector length.

Our contributions in this paper are three-fold. First, we propose a random walk model that is robust to distortion by vector length, where the probability of a word vector being generated by a discourse vector is inversely related to the angular distance between them. Second, we derive a weighting scheme from this model and compute a MAP estimate for the sentence embedding as follows: normalize the word vectors, take a weighted average of them, and then subtract from each weighted average vector the projection on their first m principal components. We call the weighting scheme derived from our random walk model *unsupervised smoothed inverse frequency* (uSIF). It is similar to SIF (Arora et al., 2017) in practice, but requires no hyperparameter tuning at all – it is completely unsupervised, allowing it to be used when there is no labelled data. Lastly, we show that our approach outperforms Arora et al.'s by up to 44.4% on textual similarity tasks, and is even competitive with state-of-the-art methods. Given the simplicity, effectiveness, and unsupervised nature of our method, we suggest it be used as a baseline for computing sentence embeddings.

2 Related Work

Word Embeddings Word embeddings are distributed representations of words, typically in a low-dimensional continuous space. These word vectors can capture semantic and lexical properties of words, even allowing some relationships to be captured algebraically (e.g., $v_{\text{Berlin}} - v_{\text{Germany}} + v_{\text{France}} \approx v_{\text{Paris}}$) (Mikolov et al., 2013). Word embeddings are generally obtained in two ways: (a) from internal representations of words in shallow neural networks (Bengio et al., 2003; Mikolov et al., 2013; Collobert and Weston, 2008); (b) from low rank approximations of co-occurrence matrices (Pennington et al., 2014).

Word Sequence Embeddings Embeddings for sequences of words (e.g., sentences) are created by composing word embeddings. This can be done simply, by doing coordinate-wise multiplication (Mitchell and Lapata, 2008) or taking an unweighted average (Mikolov et al., 2013) of the word vectors. More sophisticated architectures can also be used: for instance, recursive neural networks (Socher et al., 2011, 2013), LSTMs (Tai et al., 2015), and convolutional neural networks (Kalchbrenner et al., 2014) can be defined and trained on parse and dependency trees.

Other approaches are based on the presence of a latent vector for the entire sequence. Paragraph vectors (Le and Mikolov, 2014) are latent representations that influence the distribution of words. Skip-thought vectors (Kiros et al., 2015) are hidden representations of a neural network that encodes a sentence by trying to reconstruct its surrounding sentences. Conneau et al. (2017) leverage transfer learning by using the hidden representation of a sentence in an LSTM trained for another task, such as textual entailment. The inspiration for Arora et al. (2017) is Wieting et al. (2016b), who use word averaging after updating word embeddings by tuning them on paraphrase pairs. A later work by Wieting et al. (2017a) tried trigram-averaging and LSTM-averaging in addition to word-averaging. In that approach, vectors were tuned on the ParaNMT-50M dataset, created by using neural machine translation to translate 51M Czech-English sentence pairs into English-English pairs. This yielded state-of-the-art results on textual similarity tasks, beating the previous baseline by a wide margin.

3 Approach

3.1 The Log-Linear Random Walk Model

In Arora et al.'s original model (2016), words are generated dynamically by the random walk of a time-variant discourse vector $c_t \in \mathbb{R}^d$, representing "what is being talked about". Words are represented as $v_w \in \mathbb{R}^d$. The probability of a word w being generated at time t is given by a log-linear production model (Mnih and Hinton, 2007):

$$p(w|c_t) \propto \exp\left(\langle c_t, v_w \rangle\right) \tag{1}$$

Assuming that the discourse vector c_t does not change much over the course of the sentence, Arora et al. replace the sequence of discourse vectors $\{c_t\}$ across all time steps with a single discourse vector c_s. The MAP estimate of c_s is then the unweighted average of word vectors (ignoring any scalar multiplication).

Arora et al.'s improved random walk model (2017) allows words to also be generated: (a) by chance, with probability $\alpha \cdot p(w)$, where α is some scalar and $p(w)$ is the frequency; (b) if the word is correlated with the common discourse vector, which represents frequent discourse such as stopwords. We use c_0 to denote the common discourse vector, to be consistent with the literature. Among

other things, these changes help explain words that appear frequently despite being poorly correlated with the discourse vectors — words like *the*, for example. The probability of a word w being generated by a discourse vector c_s is then given as:

$$p(w|c_s) = \alpha \cdot p(w) + (1-\alpha) \cdot \frac{\exp(\langle \widetilde{c_s}, v_w \rangle)}{Z_{\widetilde{c_s}}},$$
$$\text{where } \widetilde{c_s} \triangleq \beta \cdot c_0 + (1-\beta) \cdot c_s, c_0 \perp c_s$$
$$Z_{\widetilde{c_s}} \triangleq \sum_{w' \in \mathcal{V}} \exp(\langle \widetilde{c_s}, v_{w'} \rangle)$$

where α, β are scalar hyperparameters, $\mathcal{V}$ is the vocabulary, $\widetilde{c_s}$ is a linear combination of the discourse and common discourse vectors parameterized by β, and $Z_{\widetilde{c_s}}$ is the partition function.

The sentence embedding for a sentence is defined as the MAP estimate of the discourse vector c_s that generated the sentence. To compute this tractably, Arora et al. (2017) assume that word vectors v_w are roughly uniformly dispersed in the latent space. This implies that the partition function $Z_{\widetilde{c_s}}$ is roughly the same for all $\widetilde{c_s}$, allowing it to be replaced with a constant Z. Assuming a uniform prior over $\widetilde{c_s}$, the maximum likelihood estimator for $\widetilde{c_s}$ on the unit sphere (ignoring normalization) is then approximately proportional to:

$$\frac{1}{|s|} \sum_{w \in s} \frac{a}{a + p(w)} \cdot v_w, \text{where } a \triangleq \frac{1-\alpha}{\alpha \cdot Z}$$

Since Z cannot be evaluated, and α is not known, a is a hyperparameter that needs tuning. This weighting scheme is called *smoothed inverse frequency* (SIF) and places a lower weight on more frequent words. The first principal component of all $\{\widetilde{c_s}\}$ in the corpus is used as the estimate for the common discourse vector c_0. The final discourse vector c_s is then produced by subtracting the projection of the weighted average on the common component (*common component removal*):

$$c_s \triangleq \widetilde{c_s} - \text{proj}_{c_0} \widetilde{c_s}$$

Arora et al. call their approach unsupervised, but others (Cer et al., 2017) have correctly noted that it is weakly supervised, since the hyperparameter a needs to be tuned on a validation set.

3.2 The Confounding Effect of Vector Length

We now propose worst-case scenarios where word vector length clearly distorts $p(s|c_s)$ due to the underlying log-linear word production model. Note that we discuss these scenarios because they are illustrative, not because they circumscribe the universe of all scenarios in which word vector length has a confounding effect.

Consider a sentence g comprising two rare words x and y, where x and y have zero similarity. Also consider some sentence h, where the only word z appears twice. g might not occur naturally, but its weighted average $\widetilde{c_g}$ would be similar to that of some longer sentence where x, y are the only non-stopwords (i.e., those with non-negligible weight). For simplicity, further assume that common component removal has negligible effect:

$$\begin{aligned} &\langle v_x, v_y \rangle = 0 \\ &c_g = \widetilde{c_g} = \frac{1}{2}\left(\frac{a}{a+p(x)} \cdot v_x + \frac{a}{a+p(y)} \cdot v_y\right) \\ &c_h = \widetilde{c_h} = \frac{a}{a+p(z)} \cdot v_z \end{aligned} \tag{2}$$

Words x, y, z are so infrequent that the probability of them being produced by chance or by the common discourse vector is negligible; the likelihood of them being produced is therefore proportional to the inner product of the discourse and word vectors. Given that the words $x, y \in g$ have zero similarity, and given that the only word $z \in h$ is identical to its discourse vector, we would expect:

$$p(h|c_h) \gg p(g|c_g) \tag{3}$$

However, (3) does not always hold. Suppose that the word embeddings lie in $\mathbb{R}^2$. Then any scalar k can be used to create a valid set of assignments for word embeddings v_x, v_y, v_z that satisfy (2):

$$v_x = \begin{bmatrix} 2k \\ 0 \end{bmatrix}, v_y = \begin{bmatrix} 0 \\ 2k \end{bmatrix}, v_z = \begin{bmatrix} k \\ k \end{bmatrix} \tag{4}$$

Assuming the words x, y, z have roughly the same frequency, they should have the same SIF-weight. Then the weighted averages, and by extension the discourse vectors (2), are the same:

$$\begin{aligned} &c_g = c_h = \frac{a}{a+p(x)} \begin{bmatrix} k \\ k \end{bmatrix} \\ &\Rightarrow \langle c_g, x \rangle = \langle c_g, y \rangle = \langle c_h, z \rangle = \frac{a}{a+p(x)} \cdot 2k^2 \\ &\Rightarrow p(g|c_g) = p(h|c_h) \end{aligned}$$

Thus it is possible for g to be generated by discourse vector c_g with roughly the same probability

as h by c_h, contradicting (3). How is this possible, given that the words in g have zero similarity with each other while those in h are identical to each other? The answer can be found in the word vector lengths. Because $||v_x||_2 = \sqrt{2}||v_z||_2$, and $p(w|c_s)$ depends on the inner product of the word and discourse vectors (1), words with longer word vectors are more likely to be produced. In fact, if v_x and v_y were multiplied by some scalar greater than 1, then $p(h|c_h)$ would be less than $p(g|c_g)$.

Generalizing Worst-Case Scenarios By manipulating the word vector length, we can also come up with a more general class of assignments that can contradict (3):

$$v_x = \begin{bmatrix} \beta k_1 \sigma \\ \beta k_2 (1-\sigma) \end{bmatrix} v_y = \begin{bmatrix} \beta k_1 (1-\sigma) \\ \beta k_2 \sigma \end{bmatrix} v_z = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{5}$$

where $\sigma \in [0,1], \beta \in \mathbb{R}, \beta \geq 2$. For convenience, we replace $\frac{a}{a+p(x)}$ with C below:

$$c_g = C \begin{bmatrix} \frac{1}{2}\beta k_1 \\ \frac{1}{2}\beta k_2 \end{bmatrix}, c_h = C \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$$

For simplicity, we assume that the words x, y, z across the two sentences are so infrequent that the probability of them being generated by chance is zero. Then the conditional probabilities of the sentences being generated are:

$$\begin{aligned} p(g|c_g) &\propto \exp\left(\langle c_g, v_x \rangle + \langle c_g, v_y \rangle\right) \\ &= \exp\left(\frac{1}{2}\beta^2 C\left({k_1}^2 + {k_2}^2\right)\right) \\ p(h|c_h) &\propto \exp\left(\langle c_h, v_z \rangle + \langle c_h, v_z \rangle\right) \\ &= \exp\left(2C\left({k_1}^2 + {k_2}^2\right)\right) \\ \therefore \beta \geq 2 &\Rightarrow p(g|c_g) \geq p(h|c_h) \end{aligned} \tag{6}$$

In this general formulation, not all scenarios are worst-case. This describes a spectrum of scenarios ranging from acceptable (e.g., $v_x = v_y = v_z$ when $\beta = 2, \sigma = 0.5$) to completely counter-intuitive (see (4)). Though these assignments only apply for word vectors in $\mathbb{R}^2$, they can easily be extended to higher-dimensional spaces.

The confound of vector length persists for longer, naturally occurring sentences. Ultimately, the underlying log-linear word production model (1) means that words with longer word vectors are more likely to be generated. Because this confound is due to model design, rather than the MLE, removing it requires redesigning the model. The exact degree of the confound varies across sentences, but in theory, it is unbounded.

3.3 An Angular Distance–Based Random Walk Model

To address the confounding effect of word vector length, we propose a random walk model where the probability of observing a word w at time t is inversely related to the angular distance between the time-variant discourse vector $c_t \in \mathbb{R}^d$ and the word vector $v_w \in \mathbb{R}^d$:

$$\begin{aligned} p(w|c_t) &\propto 1 - \frac{\arccos\left(\cos\left(v_w, c_t\right)\right)}{\pi}, \\ \text{where } \cos\left(v_w, c_t\right) &\triangleq \frac{v_w \cdot c_t}{\|v_w\|_2 \cdot \|c_t\|_2} \end{aligned} \tag{7}$$

where $\arccos\left(\cos\left(v_w, c_t\right)\right)$ is the angular distance. For the intuition behind the use of this distance metric, note that the angular distance between two vectors is equal to the geodesic distance between them on the unit sphere. Thus the angular distance can also be interpreted as the length of the shortest path between the L_2 normalized word vector and the L_2 normalized discourse vector on the unit sphere. Since the angular distance lies in $[0, \pi]$, we divide it by π to bound it in $[0,1]$. Our choice of angular distance – as opposed to, say, the exponentiated cosine similarity – is critical to avoiding hyperparameter tuning.

Assuming that the discourse vector c_t does not change much over the course of the sentence, the sequence of discourse vectors $\{c_t\}$ across all time steps can be replaced with a single discourse vector c_s for the sentence s. To model sentences more realistically, we allow words to be generated in two additional ways, as proposed in Arora et al. (2017): (a) by chance, with probability $\alpha \cdot p(w)$, where α is some scalar and $p(w)$ is the frequency; (b) if the word is correlated with one of m common discourse vectors $\{c'_m\}$, which represent various types of frequent discourse, such as stopwords. The probability of a word w being generated by discourse vector c_s is then:

$$\begin{aligned} p(w|c_s) &= \alpha \cdot p(w) + (1-\alpha) \cdot \frac{d\left(\widetilde{c_s}, v_w\right)}{Z_{\widetilde{c_s}}}, \\ \text{where } \widetilde{c_s} &\triangleq (1-\beta)\, c_s + \beta \sum_{i=1}^{m} \lambda_i\, c'_i, \;\; c_s \perp c'_i \\ d\left(\widetilde{c_s}, v_w\right) &\triangleq 1 - \frac{\arccos\left(\cos\left(v_w, c_t\right)\right)}{\pi}, \\ Z_{\widetilde{c_s}} &\triangleq \sum_{w' \in \mathcal{V}} d\left(\widetilde{c_s}, v_{w'}\right) \end{aligned} \tag{8}$$

where $\alpha, \beta, \{\lambda_i\}$ are scalar hyperparameters, $\mathcal{V}$ is the vocabulary, $\widetilde{c_s}$ is a linear combination of the

discourse and common discourse vectors parameterized by β and $\{\lambda_i\}$, and $Z_{\widetilde{c}_s}$ is the partition function. Instead of searching for the optimal hyperparameter values over some large space, as Arora et al. (2017) did, we make some simple assumptions to directly compute them.

We define the sentence embedding for some sentence s to be the MAP estimate of the discourse vector c_s that generates s. Assuming a uniform prior over possible c_s, the MAP estimate is also the MLE estimate for c_s. The log-likelihood of a sentence s is:

$$\log p(s|c_s) = \sum_{w \in s} \log p(w|c_s)$$

To maximize $\log p(s|c_s)$, we can approximate $\log p(w|c_s)$ using a first-degree Taylor polynomial:

$$f_w(\widetilde{c}_s) \triangleq \log p(w|\widetilde{c}_s)$$

$$\nabla f_w(\widetilde{c}_s) = \left[\frac{1-\alpha}{\pi \cdot Z_{\widetilde{c}_s} \cdot \exp\left(f_w(\widetilde{c}_s)\right)}\right] \frac{\frac{\partial}{\partial \widetilde{c}_s} \cos\left(v_w, \widetilde{c}_s\right)}{\sqrt{1-\cos^2\left(v_w, \widetilde{c}_s\right)}},$$

$$\frac{\partial}{\partial \widetilde{c}_s} \cos = \frac{v_w}{\|v_w\|_2 \cdot \|\widetilde{c}_s\|_2} - \cos\left(v_w, \widetilde{c}_s\right) \frac{\widetilde{c}_s}{\|\widetilde{c}_s\|_2^2}$$

Where $a \triangleq (1-\alpha)/(\alpha Z_{\widetilde{c}_s})$, C is a constant, and v'_w is a vector orthogonal to v_w with length $\|v_w\|^{-1}$:

$$\begin{aligned} f_w(\widetilde{c}_s) &\approx f_w(v'_w) + \nabla f_w(v'_w)^\intercal \left(\widetilde{c}_s - v'_w\right) \\ &= C + \frac{a}{\pi \cdot \left(p(w) + \frac{1}{2} \cdot a\right)} \cdot v_w \left(\widetilde{c}_s - v'_w\right) \\ &= C + \frac{1}{\pi}\left(\frac{a}{p(w) + \frac{1}{2} \cdot a} \langle \widetilde{c}_s, v_w \rangle\right) \end{aligned}$$

The MLE for $\widetilde{c}_s$ on the unit sphere (ignoring normalization) is then approximately proportional to:

$$\frac{1}{|s|} \sum_{w \in s} \frac{a}{p(w) + \frac{1}{2}a} \cdot v_w \tag{9}$$

The MLE of $\widetilde{c}_s$ is approximately a weighted average of word vectors, where more frequent words are down-weighted. In fact, it very closely resembles the SIF weighting scheme (Arora et al., 2017)! However, there are two key differences. For one, as we show later in this subsection, we have derived this weighting scheme from a model that is robust to the confounding effect of word vector length. Secondly, in SIF, a is a hyperparameter that needs to be tuned on a validation set. We now show that in our approach, we can calculate a directly as a function of the vocabulary $\mathcal{V}$ and the number of words in the sentence, $|s|$.

Normalization Before weighting the word vectors, we normalize them along each dimension: we construct a matrix $[v_{w_1} ... w_{w_{|s|}}]$ and take the L_2 norm of each row, which corresponds to a single dimension in $\mathbb{R}^{\mathrm{d}}$. We then multiply this d-dimensional vector element-wise with every vector in the sentence. This helps reduce the difference in variance across the dimensions.

Partition Function To calculate $Z_{\widetilde{c}_s}$, we borrow the key assumption from Arora et al. (2017) that the word vectors v_w are roughly uniformly dispersed in the latent space. Then the expected geodesic distance between a latent discourse vector and a word vector on the unit sphere is $\pi/2$, so $\mathbb{E}_{w' \in \mathcal{V}}\left[d\left(\widetilde{c}_s, v_{w'}\right)\right] = \frac{1}{2}$. Then:

$$\begin{aligned} Z_{\widetilde{c}_s} &= \sum_{w' \in \mathcal{V}} d\left(\widetilde{c}_s, v_{w'}\right) \\ &= |\mathcal{V}| \, \mathbb{E}_{w' \in \mathcal{V}}\left[d\left(\widetilde{c}_s, v_{w'}\right)\right] = \frac{1}{2}|\mathcal{V}| \end{aligned} \tag{10}$$

Odds of Random Production α is the probability that a word w will be produced by chance instead of by the discourse or common discourse vectors. To estimate α, we first consider the probability that a random word w will be produced by a discourse vector c_s at least once over n steps of a random walk:

$$p(w|c_s^1, ..., c_s^n) = 1 - \prod_{t=1}^{n}\left[1 - \frac{d(c_s^t, v_w)}{Z_{c_s}}\right]$$

$$\mathbb{E}_{w \sim \mathcal{V}}[p(w|c_s^1, ..., c_s^n)] = 1 - \left(1 - \frac{1}{|\mathcal{V}|}\right)^n$$

The number of steps taken during the random walk is itself a random variable, so we let $n = \mathbb{E}_{s \in S}|s|$. We assume that if the frequency is greater than this expectation, then the word is always produced by chance; less than this expectation, and it is always produced by the discourse or common discourse vectors. α is the proportion of the vocabulary with $p(w)$ above this threshold:

$$\alpha = \frac{\sum_{w \in \mathcal{V}} \mathbb{1}\left[p(w) > \mathbb{E}_{w \sim \mathcal{V}}[p(w|c_s^1, ..., c_s^n)]\right]}{|\mathcal{V}|} \tag{11}$$

Since we can directly calculate $Z_{\widetilde{c}_s}$ and α, we can also directly calculate $a = (1-\alpha)/(\alpha Z_{\widetilde{c}_s})$.

Common Discourse Vectors We estimate the m common discourse vectors as the first m singular vectors from the singular value decomposition of

the weighted average vectors. $\{\lambda_i\}$ are the weights on the common discourse vectors. In reality, these are unique to the word for which $p(w|c_s)$ is being evaluated. However, we let λ_i be:

$$\lambda_i = \frac{\sigma_i^2}{\sum_j^m \sigma_j^2}$$

where σ_i is the singular value for c_i'. λ_i can be interpreted as the proportion of variance explained by $\{c_1', ..., c_m'\}$ that is explained by c_i'. If removing the common discourse vectors is a form of denoising (Arora et al., 2017), increasing m, in theory, should improve results. Because the variance explained by a singular vector falls with every additional vector that is included, the choice of m is thus a trade-off between variance explained and computational cost. When $m = 1$, this is equivalent to the removal in Arora et al. (2017). We fix m at 5, since we find empirically that singular vectors beyond that do not explain much more variance. To get c_s, we subtract from $\widetilde{c_s}$ the weighted projection on each singular vector:

$$c_s \triangleq \widetilde{c_s} - \sum_{i=1}^{m} \lambda_i \operatorname{proj}_{c_i'} \widetilde{c_s}$$

We call this *piecewise common component removal*. Because our weighting scheme requires no hyperparameter tuning, it is completely unsupervised. For this reason, we call it *unsupervised smoothed inverse frequency* (uSIF). The full algorithm is given in Algorithm 1.

Note that while it is certainly *possible* to tune the hyperparameters in our model to achieve optimal results, it is not necessary to do so, which allows our method to be used when there is no labelled data. By contrast, in Arora et al.'s model (2017), hyperparameter tuning is a necessity.

Confound of Vector Length To understand why this model is not prone to the confound of word vector length, we reconsider the class of assignments for v_x, v_y, v_z in (5) and the resulting values for $\widetilde{c_g}$ and $\widetilde{c_h}$. Recall that in our example, sentence g comprises words x, y and sentence h comprises two instances of the word z. Under our new weighting scheme, C in (5) is replaced with $C' = \frac{a}{p(x)+\frac{1}{2}a}$. Note that we use $p(x)$ in C' because of the simplifying assumption that $p(x) = p(y) = p(z)$. Assuming again that $p(x) \approx 0$ and that piecewise common component removal has negligible effect,

Algorithm 1 uSIF Sentence Embedding

Input: vocabulary $\mathcal{V}$, word vectors $\{v_w : w \in \mathcal{V}\}$, frequencies $\{p(w) : w \in \mathcal{V}\}$, sentences $\mathcal{S}$
Output: sentence embeddings $\{c_s : s \in \mathcal{S}\}$

procedure EMBED
  $m \leftarrow 5$
  $n \leftarrow \mathbb{E}_{s\in\mathcal{S}}|s|$
  for all $s \in \mathcal{S}$ **do**
    $\alpha \leftarrow \frac{\sum_{w\in\mathcal{V}} \mathbb{1}\left[p(w)>1-\left(1-\frac{1}{|\mathcal{V}|}\right)^n\right]}{|\mathcal{V}|}$
    $Z \leftarrow |\mathcal{V}|/2$
    $a \leftarrow (1-\alpha)/(\alpha \cdot Z)$
    $\widetilde{c_s} \leftarrow \frac{1}{|s|}\sum_{w\in s} \frac{a}{p(w)+\frac{1}{2}a} v_w$
  end for
  $\mathcal{A} \leftarrow \left(\widetilde{c_{s_1}} \ldots \widetilde{c_{s_n}}\right)$
  for all i in 1...m **do**
    $c_i' \leftarrow i^{\text{th}}$ singular vector of $\mathcal{A}$
    $\sigma_i \leftarrow i^{\text{th}}$ singular value of $\mathcal{A}$
  end for
  for all i in 1...m **do**
    $\lambda_i \leftarrow \frac{\sigma_i^2}{\sum_j^m \sigma_j^2}$
  end for
  for all $s \in \mathcal{S}$ **do**
    $c_s \leftarrow \widetilde{c_s} - \sum_{i=1}^m \lambda_i \operatorname{proj}_{c_i'} \widetilde{c_s}$
  end for
end procedure

we can see how $p(g|c_g)$ and $p(h|c_h)$ change in our random walk model:

$$p(g|c_g) \propto \prod_{w\in\{x,y\}} \left(1 - \frac{\arccos\left(\cos\left(c_g, v_w\right)\right)}{\pi}\right)$$

$$p(h|c_h) \propto \left(1 - \frac{\arccos\left(\cos\left(c_h, v_z\right)\right)}{\pi}\right)^2 = 1$$

Because $p(g|c_g)$ is ultimately based on the cosine similarities between the discourse vector and word vectors, it is a function of the parameter $\sigma \in [0,1]$ that controls the degree of similarity between v_x and v_y. For example, for the worst-case assignments (4), $p(g|c_g) \propto 9/16$. Conversely, when $v_x = v_y = v_z$, we get $p(g|c_g) = p(h|c_h) \propto 1$. Recall that in Arora et al.'s model (2017), $\beta \geq 2$ was sufficient to ensure the counter-intuitive result of $p(g|c_g) \geq p(h|c_h)$ (6), where β was a scalar that controlled the word vector length. In contrast, in our random walk model, the effect of β – and thus the confound of vector length – is entirely absent; only the similarity between the word vectors is influential.

4 Results and Discussion

4.1 Textual Similarity Tasks

We test our approach on the SemEval semantic textual similarity (STS) tasks (2012-2015) (Agirre et al., 2012, 2013, 2014, 2015), the SemEval 2014 Relatedness task (SICK'14) (Marelli et al., 2014), and the STS Benchmark dataset (Cer et al., 2017). In these tasks, the goal is to determine the semantic similarity between a given pair of sentences; the evaluation criterion is the Pearson correlation coefficient between the predicted and actual similarity scores. To predict the similarity score, we simply encode each sentence and take the cosine similarity of their vectors. The individual scores for STS tasks are in Table 4 in the Appendix and the average scores are in Table 1. The STS benchmark scores are in Table 2. We compare our results with those from several methods, which are categorized by Cer et al. (2017) as 'unsupervised', 'weakly supervised', or 'supervised'.

4.2 Experimental Settings

For a fair comparison with Arora et al. (2017), we use the unigram probability distribution used by them, based on the enwiki dataset (Wikipedia, 3B words). Our preprocessing of the sentences is limited to tokenization. We try our method with three types of word vectors: GloVe vectors (Pennington et al., 2014), PARAGRAM-SL999 (PSL) vectors (Wieting et al., 2015), tuned on the SimLex999 dataset, and ParaNMT-50 vectors (Wieting and Gimpel, 2017a), tuned on 51M English-English sentence pairs translated from English-Czech sentence pairs. The value of n in (11) is $\mathbb{E}_{s \in \mathcal{S}}|s| \approx 11$ and was estimated using sentences from all corpora. The value of a in (9) is then 1.2×10^{-3}. Our results are denoted as **X+UP**, where $\mathbf{X} \in \{$'GloVe', 'PSL', 'ParaNMT'$\}$, **U** denotes uSIF-weighting, and **P** denotes piecewise common component removal.

4.3 Results

Our model outperforms Arora et al.'s by up to 44.4% on individual tasks (see GloVe+UP vs. GloVe+WR for the STS'12 MSRpar task in Table 4) and by up to 15.5% on yearly averages (see GloVe+UP vs. GloVe+WR for STS'12 in Table 1). Our approach proves most useful in cases where Arora et al. (2017) underperform others, such as for STS'12, where our models – GloVe+UP and PSL+UP – outperform their equivalents in Arora et al.'s results by 15.5% and 10.6% respectively. On average, our approach outperforms Arora et al.'s by around 7.6%, but the improvement is highly variable. This may be because the hyperparameter values we derived may be closer to the optima for some corpora more than others or because our other improvements – normalization and piecewise common component removal – are more effective for certain datasets.

Model	STS'12	STS'13	STS'14	STS'15	SICK14
Wieting et al. (2016b) - unsupervised					
PP	58.7	55.8	70.9	75.8	71.6
PP-XXL	61.5	58.9	73.1	77.0	72.7
tfidf-GloVe	58.7	52.1	63.8	60.6	69.4
skip-thought	30.8	24.8	31.4	31.0	49.8
Arora et al. (2017) - weakly supervised					
GloVe+WR	56.2	56.6	68.5	71.7	72.2
PSL+WR	59.5	61.8	73.5	76.3	72.9
Wieting et al.(2017b) - weakly supervised					
LSTM AVG	64.8	63.1	75.8	76.7	71.3
AVG	61.6	59.4	75.8	77.9	72.4
GRAN	62.5	63.4	75.9	77.7	72.9
Conneau et al. (2017) - unsupervised (transfer learning)					
InferSent (AllSNLI)	58.6	51.5	67.8	68.3	-
InferSent (SNLI)	57.1	50.4	66.2	65.2	-
Wieting et al. (2017a) - unsupervised					
ParaNMT Word Avg.	66.2	61.8	76.2	79.3	-
ParaNMT BiLSTM Avg.	67.4	60.3	76.4	79.7	-
ParaNMT Trigram-Word	67.8	62.7	77.4	**80.3**	-
Our Approach - unsupervised					
GloVe+UP	64.9	63.6	74.4	76.1	73.0
PSL+UP	65.8	65.2	75.9	77.6	72.3
ParaNMT+UP	**68.3**	**66.1**	**78.4**	79.0	**73.5**

Table 1: Average results (Pearson's $r \times 100$) on textual similarity tasks. The highest score in each column is in bold. "Glove+UP" is the application of uSIF-weighting (U) and piecewise common component removal (P) to GloVe word vectors; "PSL+UP' to PSL word vectors; "ParaNMT+UP", to ParaNMT word vectors.

Our best model, ParaNMT+UP, is also competitive with the state-of-the-art model, ParaNMT Trigram-Word, an average of trigram and word embeddings tuned on the ParaNMT-dataset. ParaNMT+UP outperforms ParaNMT Trigram-Word on STS'12, STS'13, and STS'14; it is narrowly outperformed on STS'15 and the STS benchmark. ParaNMT Trigram-Word's inclusion of trigram embeddings gives it an edge over our model for out-of-vocabulary words (Wieting and Gimpel, 2017a). It should be noted that ParaNMT+UP outperforms both ParaNMT Word Avg. and ParaNMT BiLSTM Avg., implying that our model composes words better than both simple averaging and BiLSTMs. Similarly, our model PSL+UP outperforms PP-XXL (Wieting et al., 2016b), despite the latter using the same word vectors and a learned projection instead.

Ablation Study On average, our weighting scheme alone is responsible for a roughly 4.4%

Unsupervised	
Doc2Vec DBOW (Le and Mikolov, 2014)	64.9
GloVe+UP	71.5
Charagram (Wieting et al., 2016a)	71.6
Paragram-Phrase (Wieting et al., 2016b)	73.2
PSL+UP	74.8
Sent2vec (Pagliardini et al., 2017)	75.5
InferSent (bi-LSTM trained on SNLI) (Conneau et al., 2017)	75.8
ParaNMT Word Avg. (Wieting and Gimpel, 2017a)	79.2
ParaNMT BiLSTM Avg. (Wieting and Gimpel, 2017a)	79.2
ParaNMT+UP	79.5
ParaNMT Trigram-Word Addition (Wieting and Gimpel, 2017a)	**79.9**
Weakly Supervised	
GloVe+WR (Arora et al., 2017)	72.0
GRAN (Wieting and Gimpel, 2017b)	76.4
Supervised	
Constituency Tree-LSTM (Tai et al., 2015)	71.9
CNN (HCTI) (Shao, 2017)	78.4

Table 2: Results (Pearson's $r \times 100$) on the STS Benchmark dataset. The highest score is in bold. The scores of our approaches are underlined.

improvement over Arora et al. The piecewise common component removal alone is responsible for a roughly 5.1% improvement, and the normalization alone is responsible for a roughly 6.7% improvement. This suggests that the benefits of our individual contributions have much overlap. The choice of tuned word vectors (e.g., ParaNMT over GloVe) can also improve results by up to 11.2%.

4.4 Supervised Tasks

We also test our approach on three supervised tasks: the SICK similarity task (SICK-R), the SICK entailment task (SICK-E), and the Stanford Sentiment Treebank (SST) binary classification task (Socher et al., 2013). To a large extent, performance on these tasks depends on the architecture that is trained with the sentence embeddings. We take the embeddings that perform best on the textual similarity tasks, ParaNMT+UP, and follow the setup in Wieting et al. (2016b). As seen in Table 3, both SIF-weighting with common component removal (Arora et al., 2017) and uSIF-weighting with piecewise common component removal (ours) perform slightly better than simple word averaging, but not as well as more sophisticated models. Past work has found that tuning the word embeddings in addition to the parameters of the model yields much better performance (Wieting et al., 2016b), as does increasing the size of the hidden layer in the classifier (Arora et al., 2017). The results here, however, suggest that regardless of such changes, our approach would not be any more effective than Arora et al.'s on these tasks. Still, our approach retains the advantage of being a completely unsupervised method that can be used when there is no labelled data.

Model	SST	SICK-R	SICK-E
ParaNMT-based (Wieting and Gimpel, 2017a)			
ParaNMT Word Avg. (300d)	80.0	83.6	80.6
ParaNMT Trigram Avg. (300d)	73.6	79.3	78.0
ParaNMT LSTM Avg. (300d)	80.6	83.9	81.9
LSTM (600d)	80.0	85.2	82.6
LSTM (900d)	81.6	86.0	83.0
BiLSTM (600d)	79.1	85.4	84.3
BiLSTM (900d)	81.3	85.8	84.4
Trigram-Word (600d, concatenation)	79.7	84.6	82.0
Trigram-Word-LSTM (900d, concatenation)	82.0	85.4	83.8
BILSTM AVG (4096)	82.8	85.9	83.8
ParaNMT+WR† (Arora et al., 2017)	80.5	83.9	80.9
ParaNMT+UP† (ours)	80.7	83.8	81.1
Other Approaches			
BiLSTM-Max (on AllNLI) (Conneau et al., 2017)	84.6	**88.4**	**86.3**
skip-thought (Kiros et al., 2015)	82.0	85.8	82.3
BYTE mLSTM (Radford et al., 2017)	**91.8**	79.2	-

Table 3: Results on the SST, SICK-R, and SICK-E tasks. The best score for each task is bolded. † indicates our implementation.

5 Future Work

There are several possibilities for future work. For one, the values we derived for $Z_{\tilde{c}_s}, \alpha, a$ and $\{\lambda_i\}$ are not necessarily optimal. While they are based on reasonable assumptions, there are likely sentence-specific and task-specific values that yield better results. Hyperparameter search is one way of finding these values, but that would require supervision. It may be possible, however, to theoretically derive more optimal values.

6 Conclusion

We first showed that word vector length has a confounding effect on the log-linear random walk model of generating text (Arora et al., 2017), the basis of a strong baseline method for sentence embeddings. We then proposed an angular distance–based random walk model where the probability of a sentence being generated is robust to distortion from word vector length. From this model, we derived a simple approach for creating sentence embeddings: normalize the word vectors, compute a weighted average, and then modify it using SVD. Unlike in Arora et al., our approach does not require hyperparameter tuning – it is completely unsupervised and can therefore be used when there is no labelled data. Our approach outperforms Arora et al.'s by up to 44.4% on textual similarity tasks and is even competitive with state-of-the-art methods. Because our simple approach is tough-to-beat, robust, and unsupervised, it is an ideal baseline for computing sentence embeddings.

Acknowledgments

We thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for financial support. We thank John Wieting for providing pre-trained ParaNMT word embeddings and Graeme Hirst for his many insightful suggestions.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings SemEval@ NAACL-HLT*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings SemEval@ COLING*, pages 81–91.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. Sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. 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*, pages 385–393. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to PMI-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

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

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

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.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 104–113.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings SemEval@ COLING*, pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 236–244.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648. ACM.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Yang Shao. 2017. Hcti at semeval-2017 task 1: Use convolutional neural network to evaluate semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 130–133.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

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

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

John Wieting and Kevin Gimpel. 2017a. Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. *arXiv preprint arXiv:1711.05732*.

John Wieting and Kevin Gimpel. 2017b. Revisiting recurrent networks for paraphrastic sentence embeddings. *arXiv preprint arXiv:1705.00364*.