Word Rotator's Distance

Sho Yokoi [1,2] **Ryo Takahashi** [1,2] **Reina Akama** [1,2] **Jun Suzuki** [1,2] **Kentaro Inui** [1,2]
[1] Tohoku University [2] RIKEN
{yokoi, ryo.t, reina.a, jun.suzuki, inui}@ecei.tohoku.ac.jp

Abstract

One key principle for assessing textual similarity is measuring the degree of semantic overlap between two texts by considering the word alignment. Such alignment-based approaches are both intuitive and interpretable; however, they are empirically inferior to the simple cosine similarity between general-purpose sentence vectors. To remedy this, we focus on the fact that the *norm* of word vectors is a good proxy for word importance, and the *angle* of them is a good proxy for word similarity. Alignment-based approaches do not distinguish the norm and direction, whereas sentence-vector approaches automatically use the norm as the word importance. Accordingly, we propose to decouple word vectors into their norm and direction then computing the alignment-based similarity using earth mover's distance (optimal transport), which we refer to as *word rotator's distance.* Furthermore, we demonstrate how to "grow" the norm and direction of word vectors (*vector converter*); this is a new systematic approach derived from the sentence-vector estimation methods, which can significantly improve the performance of the proposed method. On several STS benchmarks, our simple proposed methods outperformed not only alignment-based approaches but also strong baselines. [1]

1 Introduction

This paper addresses the task of semantic textual similarity (STS), the goal of which is to measure the degree of semantic equivalence between two sentences (Agirre et al., 2012). High-quality STS methods can be used to upgrade loss functions and automatic evaluation metrics of text generation tasks because one of the requirements of these metrics is precisely the calculation of STS (Wieting et al., 2019; Zhao et al., 2019; Zhang et al., 2019).

There are two major approaches to tackling STS. One is to measure the degree of semantic overlap between the texts by considering the word alignment, which we refer to as *alignment-based approaches* (Sultan et al., 2014; Kusner et al., 2015; Zhao et al., 2019). The other is to generate general-purpose sentence vectors from two texts (typically composed of word vectors) and then calculate their similarity, which we refer to as *sentence-vector approaches* (Arora et al., 2017; Ethayarajh, 2018). Alignment-based approaches are consistent with human intuitions concerning textual similarity, and their predictions are interpretable; however, the performance of such approaches is lower than that of sentence-vector approaches.

We hypothesize that one reason for the inferiority of alignment-based approaches is that they do not separate the *norm* and *direction* of the word vectors. Conversely, sentence-vector approaches automatically exploit the norm of word vectors as the relative importance of words.

Accordingly, we propose a new STS method that first decouples word vectors into their norms and direction vectors, then aligns the direction vectors using earth mover's distance. The key idea is to map the *norm* and *angle* of word vectors to the EMD parameters *probability mass* and *transportation cost*, respectively. The proposed method is natural from both perspectives of optimal transport and word embeddings, preserves the features of alignment-based methods, and can directly incorporate the sentence-vector estimation methods, leading to fairly high performance.

Our main contributions are as follows.

- We demonstrate that the norm of a word vector implicitly encodes the weight of a word and that the angle between word vectors is a good proxy for the dissimilarity of the words.
- We propose a new textual similarity measure, word rotator's distance, which can separately uti-

[1] The source code is avaliable at https://github.com/eumesy/wrd

lize the norm and direction of word vectors.

- To enhance the method, we further propose a new word-vector conversion mechanism with the help of recent methods of the recent sentence-vector estimation methods.
- We demonstrate that the proposed methods achieve high performance compared to strong baseline methods on several STS tasks.

2 Task and Notation

Semantic textual similarity (STS) is the task of measuring the degree of semantic equivalence between two sentences (Agirre et al., 2012). For example, the sentences "*Two boys on a couch are playing video games.*" and "*Two boys are playing a video game.*" are mostly equivalent (the similarity score of 4 out of 5) while the sentences "*The woman is playing the violin.*" and "*The young lady enjoys listening to the guitar.*" are not equivalent but on the same topic (score of 1) (Agirre et al., 2013). System predictions are customarily evaluated by Pearson correlation with the gold scores. Hence, systems are only required to predict relative similarity rather than absolute scores.

We focus on *unsupervised* STS, following Arora et al. (2017) and Ethayarajh (2018). That is, we utilize only pre-trained word vectors, and do not use any supervision including training data for related tasks (e.g., natural language inference) and external resources (e.g., paraphrase database). *Semi-supervised* methods that utilize such external corpora have been successful in English STS. However, the need for external corpora is a major obstacle when applying STS, a fundamental technology, to low-resource languages.

Formally, given sentences s and s' consisting of n and n' words from the vocabulary $\mathcal{V}$

$$s = (w_1, \dots, w_n),\ s' = (w'_1, \dots, w'_{n'}), \qquad (1)$$

the goal is to predict the similarity $\mathrm{sim}(s, s') \in \mathbb{R}$. Bold face $\boldsymbol{w}_i \in \mathbb{R}^d$ denotes the word vector corresponding to word w_i. Let $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ denote the dot product and the Euclidean norm, respectively

$$\langle \boldsymbol{w}, \boldsymbol{w}' \rangle := \boldsymbol{w}^\top \boldsymbol{w}', \quad \|\boldsymbol{w}\| := \sqrt{\langle \boldsymbol{w}, \boldsymbol{w} \rangle}. \qquad (2)$$

3 Related Work

We briefly review the methods that are directly related to unsupervised STS.

Alignment-based Approach. One major approach for unsupervised STS is to compute the degree of semantic overlap between two texts (Sultan et al., 2014, 2015). Recently, determining the soft alignment between word vector sets has become a mainstream method. Tools used for alignment include attention mechanism (Zhang et al., 2019), fuzzy set (Zhelezniak et al., 2019), and earth mover's distance (EMD) (Kusner et al., 2015; Clark et al., 2019; Zhao et al., 2019).

Of those, EMD has several unique advantages. First, it has a rich theoretical foundation for measuring the differences between probability distributions in a metric space (Villani, 2009; Peyré and Cuturi, 2019). Second, EMD can incorporate structural information such as syntax trees (Alvarez-Melis et al., 2018; Titouan et al., 2019). Finally, with a simple modification, EMD can be differentiable and can be incorporated into larger neural networks (Cuturi, 2013). Despite these advantages, EMD-based methods have underperformed sentence-vector-based methods on STS tasks. The goal of this study is to identify and resolve the obstacles faced by EMD-based methods (Section 5).

Sentence-vector Approach. Another popular approach is to employ general-purpose sentence vectors of given texts and to compute the cosine similarity between such vectors. A variety of methods to compute sentence vectors have been proposed, ranging from deep sentence encoders (Kiros et al., 2015; Conneau et al., 2017; Cer et al., 2018), learning and using word vectors optimized for summation (Khodak et al., 2018; Pagliardini et al., 2018; Wieting and Gimpel, 2018), to estimating latent sentence vectors from pre-trained word vectors (Arora et al., 2017; Ethayarajh, 2018; Liu et al., 2019b). This paper demonstrates that some recently proposed sentence vectors can be reformulated as a sum of the converted word vectors. By using the converted word vectors, our method can achieve similar or better performance compared to sentence-vector approaches (Section 6).

4 Word Mover's Distance and its Issue

4.1 Earth Mover's Distance

Intuitively, *earth mover's distance (EMD)*[2] (Villani, 2009; Santambrogio, 2015; Peyré and Cuturi, 2019) is the minimum cost required to turn one pile of dirt

[2] In this paper, following convention, we use the term earth mover's distance in the sense of optimal transport cost according to the Kantrovich formulation. If the cost is a distance, it can also be called the 1-Wasserstein distance.

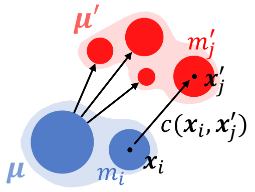

Figure 1:
Earth Mover's Distance.

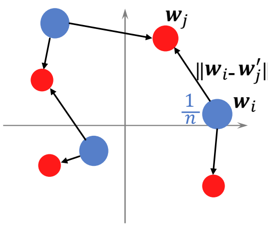

Figure 2:
Word Mover's Distance.

into the another pile of dirt (Figure 1). Formally, EMD takes the following inputs.

1. Two **probability distributions**, μ (initial arrangement) and μ' (final arrangement)[3]:

$$\mu = \Big\{(x_i,\, m_i)\Big\}_{i=1}^{n},\ \mu' = \Big\{(x'_j,\, m'_j)\Big\}_{j=1}^{n'}. \tag{3}$$

   Here, μ denotes a probability distribution, in which each point $x_i \in \mathbb{R}^d$ has a probability mass $m_i \in [0, 1]$ ($\sum_i m_i = 1$). In Figure 1, each circle represents a pair (x_i, m_i), where the location and size of the circle represent a vector x_i and its probability m_i, respectively.

2. The **transportation cost function**, c:

$$c\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}. \tag{4}$$

   Here, $c(x_i, x'_j)$ determines the transportation cost per unit amount (distance) between two points x_i and x'_j.

The EMD between μ and μ' is then defined via the following optimization problem:

$$\mathrm{EMD}(\mu, \mu'; c) := \min_{T \in \mathbb{R}_{\geq 0}^{n \times n'}} \sum_{i,j} T_{ij}\, c(x_i, x'_j), \tag{5}$$

$$\text{s.t.} \begin{cases} T\mathbb{1}_n = m := (m_1, \ldots, m_n)^\top, \\ T^\top \mathbb{1}_{n'} = m' := (m'_1, \ldots, m'_{n'})^\top. \end{cases} \tag{6}$$

A solution $T \in \mathbb{R}_{\geq 0}^{n \times n'}$ denotes a transportation plan, in which each element T_{ij} represents the mass transported from x_i to x'_j. To summarize, $\mathrm{EMD}(\mu, \mu'; c)$ is the cost of the best transportation plan between two distributions μ and μ'.

Side Benefit: Alignment. Under the above optimization, if the locations x_i and x'_j are *close* (i.e., if the transportation cost $c(x_i, x'_j)$ is small), they are likely to be *aligned* (i.e., T_{ij} might be assigned a large value). In this way, EMD can be seen as aligning the points of two discrete distributions. This is one of the reasons why we adopt EMD as a key technology for the computation of STS.

4.2 Word Mover's Distance

Word mover's distance (WMD) (Kusner et al., 2015) is a dissimilarity measure between texts, and is a pioneering work that introduced EMD to the natural language processing (NLP) community. Our study is strongly inspired by this work. We introduce WMD in preparation for our proposed method.

WMD is the cost of transporting a set of word vectors in the embedding space (Euclidean space) (Figure 2). Formally, after removing stopwords, Kusner et al. (2015) regard each sentence s as a uniformly weighted distribution μ_s consisting of word vectors (bag-of-word-vectors distribution):

$$\mu_s := \Big\{\Big(w_i,\ \frac{1}{n}\Big)\Big\}_{i=1}^{n},\ \mu_{s'} := \Big\{\Big(w'_j,\ \frac{1}{n'}\Big)\Big\}_{j=1}^{n'}. \tag{7}$$

In Figure 2, each circle represents each word, where the location and size of the circle represent the vector w_i and its weight $\frac{1}{n}$, respectively. Next, they use Euclidean distance as the transportation cost between the word vectors

$$c_{\mathrm{E}}(w_i, w'_j) := \|w_i - w'_j\|. \tag{8}$$

Then, WMD is defined as the EMD between two such distributions using the cost function c_{E}

$$\mathrm{WMD}(s, s') := \mathrm{EMD}(\mu_s, \mu_{s'}; c_{\mathrm{E}}). \tag{9}$$

4.3 Issues with Word Mover's Distance

Despite its intuitive formulation, WMD would often misalign words with each other, and the STS performance of WMD is lower than that of methods that simply add up word vectors and the compute cosine similarity. For example, by WMD, "noodle" and "snack" might be aligned instead of "noodle" and "pho" (a type of Vietnamese noodle).

5 Word Rotator's Distance

In this section, we first discuss the role of the norm and direction of word vectors. Then, we describe the issues with WMD from the perspective of the role of the norm and direction. Finally, we propose our new method, word rotator's distance, which is able to resolve these issues.

[3] Strictly speaking, Equation 3 is $\mu = \sum_{i=1}^{n} m_i \delta[x_i]$, where the Dirac delta function describes a discrete probability measure. In this paper, we omit delta for notational simplicity.

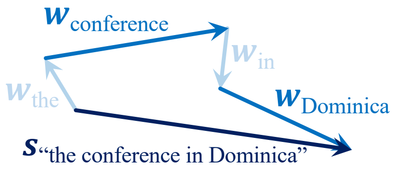

Figure 3: The operation of addition implicitly uses the norm of the vectors as the weighting factor.

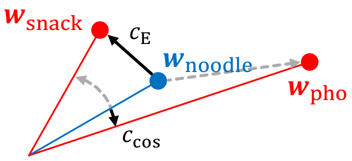

Figure 4: Euclidean distance "mixes up" the norm (a weighting factor for each word) and direction vectors (for word dissimilarity).

5.1 Role of Norm and Direction

We hypothesis that the norm and direction of word vectors have the following different roles.

- **Norm of a word vector as weighting factor**: The norm of a word vector indicates the extent to which the word contributes to the overall meaning of a sentence.
- **Angle between word vectors as dissimilarity**: The angle between two word vectors (the difference between the direction of these vectors) indicates the (dis)similarity of the two words.

We elaborate on the validity of this hypothesis in this section. Henceforth, λ_i and $\boldsymbol{u}_i$ denote the norm and the direction vector of word vector $\boldsymbol{w}_i$, resp.:

$$\lambda_i := \|\boldsymbol{w}_i\|, \;\; \boldsymbol{u}_i := \boldsymbol{w}_i/\lambda_i \;\; \big(\boldsymbol{w}_i = \lambda_i \boldsymbol{u}_i\big). \quad (10)$$

Each $\boldsymbol{u}_i$ is a unit vector $\big(\|\boldsymbol{u}_i\| = 1\big)$.

Additive Compositionality. As a starting point, we review the well-known nature of additive compositionality. The NLP community has confirmed that a simple sentence vector, i.e., the average of the vectors of the words in a sentence, can achieve remarkable results when assessing STS tasks, as well as many downstream tasks (Mitchell and Lapata, 2010; Mikolov et al., 2013; Wieting et al., 2016; Perone et al., 2018; Ma et al., 2019).

$$\boldsymbol{s}_{\text{ADD}} = \frac{1}{n}\sum_{w_i \in s} \boldsymbol{w}_i, \quad \boldsymbol{s}'_{\text{ADD}} = \frac{1}{n'}\sum_{w'_j \in s'} \boldsymbol{w}'_j, \quad (11)$$

$$\operatorname{sim}(s, s') = \cos(\boldsymbol{s}_{\text{ADD}}, \boldsymbol{s}'_{\text{ADD}}). \quad (12)$$

Norm as Weighting Factor. At first glance, Equation 11 may appear to treat each word vector equally. However, several studies have confirmed that the norm of word vectors has a large dispersion (Schakel and Wilson, 2015; Arefyev et al., 2018). In other words, a sentence vector would contain word vectors of various sizes. In such a situation, a word vector having a large norm will dominate in the resulting sentence vector, and vice versa (Figure 3). Here, the usefulness of additive composition (implicit weighting by the norm) suggests that *the norm of each word vector functions as the weighting factor of the word when generating a sentence representation*. In our experiments, we provide data-driven evidence to support this claim.

Moreover, regarding the relationship between the word vector norm and the word importance, the followings are known: (i) content words tend to have larger norms than function words (Schakel and Wilson, 2015); and (ii) fine-tuned word vectors have larger norms for medium-frequency words, which is consistent with the traditional weighting guideline by Luhn in information retrieval (Khodak et al., 2018; Pagliardini et al., 2018). Both suggest that the norm serves as a weighting factor in situations where additive composition works.

Angle as Dissimilarity. What does a direction vector (i.e., the rest of the word vector "minus" its norm) represent?[4] Obviously, the most common calculation using the direction vectors of words is to measure their angles, i.e., their cosine similarity

$$\cos(\boldsymbol{w}, \boldsymbol{w}') = \frac{\langle \boldsymbol{w}, \boldsymbol{w}' \rangle}{\lambda \lambda'} = \langle \boldsymbol{u}, \boldsymbol{u}' \rangle. \quad (13)$$

It is widely known that the cosine similarity of word vectors trained on the basis of the distributional hypothesis approximates word similarity well (Pennington et al., 2014; Mikolov et al., 2013; Bojanowski et al., 2017). Naturally, *the difference in direction vectors represents the dissimilarity of words.* In our experiments, we confirm that cosine similarity is an empirically better proxy for word similarity compared to other measures.

5.2 Why doesn't WMD Work?

According to the above discussion, WMD has the following limitations.

- **Weighting of words:** While EMD can consider the weights of each point via their probability

[4] Analogous to the polar coordinate system, Equation 10 decouples each word vector into a one-dimensional norm and a $(d-1)$-dimensional direction vector.

	noodle	*pho*	*snack*	*Pringles*
noodle	-	**0.43**	0.58	0.83
pho	**0.43**	-	0.73	0.94
snack	0.58	0.73	-	**0.56**
Pringles	0.83	0.94	**0.56**	-

(a) Cosine distance.

	noodle	*pho*	*snack*	*Pringles*
noodle	-	3.52	$\underline{\mathbf{3.39}}$	4.62
pho	**3.52**	-	4.52	5.60
snack	$\underline{\mathbf{3.39}}$	4.52	-	3.84
Pringles	4.62	5.60	**3.84**	-

(b) Euclidean distance.

Table 1: Differences in behavior between cosine and Euclidean distance. For each row, the lowest value (the closest word) is shown in **bold**. Inappropriate alignments are further **underlined**. We used pre-trained word2vec (Mikolov et al., 2013) as the word vectors.

mass (3) and the weighting factor of each word is encoded in the norm, WMD ignores the norm and weights each word vector uniformly (7).

- **Dissimilarity between words:** While EMD can consider the distance between points via a transportation cost (4) and the dissimilarity between words can be measured by angle, WMD uses Euclidean distance, which *mixes* the weighting factor and the dissimilarity.

The reason why the *mixing* is problematic can also be explained by the following. Euclidean transportation cost (8) would misestimate the similarity of word pairs as low$_{\langle \mathrm{A} \rangle}$, whose meanings are close$_{\langle \mathrm{B} \rangle}$ but whose concreteness or importance is very different$_{\langle \mathrm{C} \rangle}$, such as "noodle" and "pho" (Figure 4). This is clear from the relationship between the Euclidean (8) and the cosine distance (14).

$$c_{\cos}(\boldsymbol{w}, \boldsymbol{w}') := 1 - \cos(\boldsymbol{w}, \boldsymbol{w}') \tag{14}$$

$$c_{\mathrm{E}}(\boldsymbol{w}, \boldsymbol{w}') = \sqrt{(\lambda \boldsymbol{u} - \lambda' \boldsymbol{u}')^{\top} (\lambda \boldsymbol{u} - \lambda' \boldsymbol{u}')} \tag{15}$$

$$= \sqrt{\lambda \lambda' \left(2 c_{\cos}(\boldsymbol{w}, \boldsymbol{w}') + (\lambda - \lambda')^2\right)}. \tag{16}$$

From Equation 16, $c_{\mathrm{E}}(\boldsymbol{w}, \boldsymbol{w}')$ would be estimated as being large$_{\langle \mathrm{A} \rangle}$ even if $c_{\cos}(\boldsymbol{w}, \boldsymbol{w}')$ is small$_{\langle \mathrm{B} \rangle}$, as long as $|\lambda - \lambda'|$ is large$_{\langle \mathrm{C} \rangle}$. This undesirable property is also confirmed when using real data. Table 1 and Figure 4 show the cosine and Euclidean distances between the vectors of "noodle," "pho," "snack," and "Pringles" (the name of a snack). By using Euclidean distance, "noodle" and "snack" are judged to be similar (more likely to be aligned) than "noodle" and "pho".

5.3 Word Rotator's Distance

Given the above considerations, we propose a simple yet powerful sentence similarity measure using EMD. Our method regards each sentence as a discrete distribution on the unit hypersphere and calculates EMD on this hypersphere (Figure 5). Because the alignment of the direction vectors corresponds to a rotation on the unit hypersphere, we refer to the method as *word rotator's distance (WRD)*.

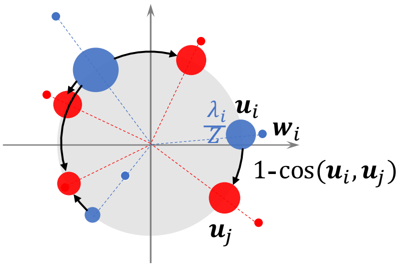

Figure 5: Word Rotator's Distance.

Formally, we regard each sentence s as a discrete distribution $\boldsymbol{\nu}_s$ consisting of direction vectors weighted by their norm (bag-of-direction-vectors distribution)

$$\boldsymbol{\nu}_s := \left\{ \left(\boldsymbol{u}_i, \frac{\lambda_i}{Z} \right) \right\}_{i=1}^{n}, \ \boldsymbol{\nu}_{s'} := \left\{ \left(\boldsymbol{u}'_j, \frac{\lambda_j}{Z'} \right) \right\}_{j=1}^{n'}, \tag{17}$$

where Z and Z' are normalizing constants ($Z := \sum_i \lambda_i$, so as Z'). In Figure 5, each circle represents a word, where the location and size of the circle represent the direction vector $\boldsymbol{u}_i$ and its weight λ_i/Z, respectively. For the cost function, we use the cosine distance.

$$c_{\cos}(\boldsymbol{u}_i, \boldsymbol{u}'_j) = 1 - \cos(\boldsymbol{u}_i, \boldsymbol{u}'_j) \tag{18}$$

That is, to align words, it takes a cost of rotation. Then, the WRD between two sentences is:

$$\mathrm{WRD}(s, s') := \mathrm{EMD}(\boldsymbol{\nu}_s, \boldsymbol{\nu}_{s'}; c_{\cos}). \tag{19}$$

Unlike WMD, the above procedure allows our WRD to follow appropriate correspondences between EMD and the word vectors.

- Probability mass (**weight** of each point) ↔ Norm (**weight** of each word)
- Transportation cost (**distance** between points) ↔ Angle (**dissimilarity** between words)

Algorithm. To ensure reproducibility, the specific (and quite simple) algorithm and implementation guidelines for WRD are shown in Appendix C.

6 Vector Converter-enhanced WRD

To further improve the performance of WRD, we attempted to integrate methods for estimating latent sentence vectors, the most powerful sentence encoders for STS, into WRD. However, determining a method to combine sentence-vector estimation methods with WRD is not a straightforward task. This is because WRD takes *word vectors* as input, whereas sentence-vector estimation methods require the processing of *sentence vectors*.

6.1 From Sentence Vector to Word Vector

Sentence-vector Estimation. On the basis of Arora's pioneering random-walk language model (LM) (Arora et al., 2016, 2017), a number of sentence-vector estimation methods have been proposed (Arora et al., 2017; Ethayarajh, 2018; Liu et al., 2019a,b) and achieved great success in many NLP applications, including STS. Given pre-trained vectors of the words that compose a sentence, these methods allow us to estimate the latent sentence vectors that generated the word vectors.

Such sentence-encoder methods can be summarized in the following form

$$\mathrm{Encode}(s) = f_3\Biggl(\frac{1}{n}\sum_{w\in s}\alpha_2(w)f_1(\boldsymbol{w})\Biggr), \qquad (20)$$

where

- $f_1\colon \mathbb{R}^D \to \mathbb{R}^D$ "denoises" each word vector,
- $\alpha_2\colon \mathcal{V} \to \mathbb{R}$ scales each word vector, and
- $f_3\colon \mathbb{R}^D \to \mathbb{R}^D$ "denoises" each sentence vector.

Here, we focus only on the form of the equation for sentence-vector estimation. For specific algorithms, see the experimental section and Appendix D.

Word Vector Converter. By noticing that all the proposed denoising function f_3 is linear, Equation 20 can be rewritten as

$$\mathrm{Encode}(s) = \frac{1}{n}\sum_{w\in s}\widetilde{\boldsymbol{w}} \qquad (21)$$

$$\widetilde{\boldsymbol{w}} = f_{\mathrm{VC}}(\boldsymbol{w}) := f_3\left(\alpha_2(w)\cdot f_1(\boldsymbol{w})\right). \qquad (22)$$

That is, the encoders first perform a transformation f_{VC} on each word vector independently and then sum them up (additive composition!). We refer to the f_{VC} as *(word) vector converter (VC)*.

6.2 Norm and Direction

We believe that the vector converter $f_{\mathrm{VC}}\colon \boldsymbol{w} \mapsto \widetilde{\boldsymbol{w}}$ improves the norm and direction of pre-trained word vectors for the reasons discussed in the following sections. We will conduct our experiments to support this hypothesis.

Norm as Weighting Factor. In Section 5, in light of the success of additive composition, we proposed the use of norms to weight words. In Section 6.1, we confirmed that the sentence-vector estimation methods, which have achieved greater success in STS than standard additive composition, simply add up the transformed word vectors (improved additive composition). Therefore, we expect that the importance of a word w is "better" encoded in the norm of a converted word vector $\widetilde{\boldsymbol{w}}$ than in that of the original word vector $\boldsymbol{w}$.

Angle as Dissimilarity. The denoising function f_1, on the basis of the random-walk LM, makes the word vector space isotropic (i.e., uniform in the sense of angle); as a result, the angle of word vectors becomes a better proxy for the word dissimilarity (Mu and Viswanath, 2018; Liu et al., 2019b). Further, the conversion by α_2 and f_3 allows a more realistic generative model to be assumed (i.e., the unigram probability is taken into account). This would improve the isotropy of the vector space, making the angle of the word vectors would be a better proxy for the dissimilarity of words.

6.3 Vector Converter-enhanced WRD

As we have shown so far, converted word vectors $\{\widetilde{\boldsymbol{w}}\}$ may have preferred properties in terms of their norm and direction. In addition, because they remain word vectors, $\{\widetilde{\boldsymbol{w}}\}$ can be used as is for the input of WRD. Let $\widetilde{\lambda}$ and $\widetilde{\boldsymbol{u}}$ denote the norm and direction vector of $\widetilde{\boldsymbol{w}}$, respectively, a variant of WRD using $\{\widetilde{\boldsymbol{w}}\}$ is

$$\widetilde{\boldsymbol{\nu}}_s := \Bigl\{\bigl(\widetilde{\boldsymbol{u}}_i,\ \frac{\widetilde{\lambda}_i}{Z}\bigr)\Bigr\}_{i=1}^{n}, \widetilde{\boldsymbol{\nu}}_{s'} := \Bigl\{\bigl(\widetilde{\boldsymbol{u}}'_j,\ \frac{\widetilde{\lambda}_j}{Z'}\bigr)\Bigr\}_{j=1}^{n'}, \qquad (23)$$

$$\mathrm{WRD}_{\text{with VC}}(s, s') := \mathrm{EMD}(\widetilde{\boldsymbol{\nu}}_s, \widetilde{\boldsymbol{\nu}}_{s'}; c_{\cos}), \qquad (24)$$

where Z and Z' are normalizing constants. We believe that using $\{\widetilde{\boldsymbol{w}}\}$ will improve the performance of WRD because WRD depends on the weights and dissimilarities encoded in the norm and angle.

7 Experiments

As word vectors, we mainly used the most standard **GloVe** (Pennington et al., 2014), **word2vec** (Mikolov et al., 2013), and **fastText** (Bojanowski et al., 2017). Note that, as of now, contextualized word embeddings have not been effective in unsupervised STS; we elaborate on it in Appendix B. As STS datasets, we mainly used **STS'15** (Agirre et al., 2015) for comparison with Zhelezniak et al. (2019); Kiros et al. (2015); Peters et al. (2018); **STS-B** (Cer et al., 2017),

	ADD	W/O NORM
GloVe	**54.16**	46.25
word2vec	**72.43**	63.20
fastText	**70.40**	56.31
ELMo	**63.22**	57.96
BERT-large	**65.76**	64.04

(a) Pre-trained word vectors.

	ADD	W/O NORM
GloVe	**54.16**	46.25
+ A	**68.30**	59.62
+ AW	**76.68**	59.62
+ **VC**(AWR)	**79.13**	63.60

(b) Converted word vectors.

Table 2: The effect of the norm on additive composition. **Pearson's $r \times 100$** between the predicted scores and the gold scores for each word vector (each row) and each method (each column). The STS-B dataset (dev) is used. The best result in each row is indicated in **bold**.

	COS GloVe	L2 GloVe	DOT GloVe
MEN	**80.49**	73.36	**80.79**
MTurk287	**69.18**	60.87	**69.50**
MC30	**78.81**	75.22	76.77
RW	**47.28**	40.37	45.64
RG65	76.90	70.75	**77.79**
SCWS	**62.96**	55.87	61.94
SimLex999	**40.84**	35.16	38.99
WS353-REL	68.75	49.74	**72.35**
WS353-SIM	**79.57**	69.03	**79.54**

(a) Which measure of word similarity should be used?

	COS GloVe	COS + A	COS + **VC** (AWR)
MEN	80.49	**82.43**	**82.26**
MTurk287	69.18	**72.77**	69.32
MC30	78.81	77.99	**80.67**
RW	47.28	**54.75**	**54.34**
RG65	**76.90**	75.18	**76.89**
SCWS	62.96	**67.09**	65.83
SimLex999	40.84	46.74	**49.83**
WS353-REL	68.75	70.73	**72.35**
WS353-SIM	79.57	**80.97**	79.32

(b) VC gradually "grow" the direction of word vectors.

Table 3: Spearman's $\rho \times 100$ between the predicted scores and the gold scores is reported. In each row, the best result and the results where the difference from the best result is < 0.5 are indicated in **bold**. "+ AW" is omitted from Table 3b because W (scaling function) alone does not change the angle.

one of the most actively used datasets, and **Twitter'15** (Xu et al., 2015) to validate methods against casual writing. See Appendix A for the complete list. For VC, we used the followings algorithms.

- f_1: **A**ll-but-the-top (Mu and Viswanath, 2018), conceptor **N**egation (Liu et al., 2019a), norm-based feature **S**caling (Ethayarajh, 2018)[5].
- α_2: SIF **W**eighting (Arora et al., 2017), **U**nsupervised SIF weighting (Ethayarajh, 2018).
- f_3: common component **R**emoval (Arora et al., 2017), **P**iecewise common component removal (Ethayarajh, 2018), **C**onceptor removal (Liu et al., 2019b).

Henceforth, a bold character denotes each method. In addition, **VC**(AWR), for example, denotes the **V**ector **C**onverter induced by **A**, **W**, and **R**. Note that we did not tune the hyperparameters; instead, we used the values acquired from previous studies. See Appendix D for detailed settings. All experiments were performed on a laptop computer with 2.2 GHz 6-core Intel Core i7 and 32 GB of RAM.

7.1 Workings of Norm

We hypothesized that the norm functioned as a weighting factor because additive composition (11) implicitly uses the norm to weight words.

Pre-trained Word Vectors. Let us consider another additive composition that excludes the effect of weighting by the norm

$$s_{\text{ADD W/O NORM}} = \sum_{w_i \in s} \boldsymbol{w}_i/\lambda_i = \sum_{w_i \in s} \boldsymbol{u}_i. \qquad (25)$$

Table 2a shows the experimental results on the STS-benchmark dataset using two types of sentence vectors (11, 25). According to these results, ignoring the norm of the word vectors leads to consistently poor performances. This demonstrates that the norm of a word vector plays the role of the weighting factor of the word.

Converted Word Vectors. In addition, to verify our hypothesis that VC improves the norm, we performed the above experiment on the converted word vector. Table 2b shows that ignoring the norm of a word vector results in consistently worse predictive performances. Even when the norm is ignored, the performance is improved by the transformation sequence of the word vectors. The reason for this might be the improvement in the direction vector (for the word dissimilarity).

7.2 Workings of Angle

We assumed that the angle between two word vectors is a good proxy for the dissimilarity of two words. Presently, the cosine similarity between word vectors is one of the most widely used metrics to compute word dissimilarity. However, several alignment-based STS methods employ Euclidean distance (Kusner et al., 2015) or dot product (Zhelezniak et al., 2019). Therefore, the question arises as to which is the most suitable method for computing word dissimilarity. Here, we experimentally confirm the superiority of cosine similarity via nine word-similarity tasks. See Appendix A for the details of the datasets.

[5] Ethayarajh (2018) proposed three methods: S, U, and P. For the sake of correctness, we abbreviate the series of methods as SUP, while abbreviated as UP in the original paper.

	WMD	**WRD**	WMD	**WRD**
Removing Stopwords			✓	✓
GloVe	62.56	**64.66**	**71.34**	**71.13**
GloVe + A	65.74	**68.83**	**75.19**	**75.19**
GloVe + AW	63.34	**77.21**	74.41	**76.44**
GloVe + A + SIF weights	76.81	-	76.56	-
GloVe + **VC**(AWR)	61.42	**79.20**	72.81	**78.60**

Table 4: **Pearson's r × 100** between the predicted scores and the gold scores for each word vector (each row) and each method (each column). The STS-B dataset (dev) is used. The best result and results where the difference from the best < 0.5 in each row are in **bold**, and the best result are further **underlined**.

	STS'15	STS-B	Twitter
GloVe – Additive Composition			
GloVe[†]	56.08	45.57	29.35
GloVe + WR[†] (Arora et al., 2017)	67.74	62.85	40.03
GloVe + SUP[†] (Ethayarajh, 2018)	**74.38**	**71.03**	**50.24**
GloVe – Considering Word Alignment			
WMD[†] (Kusner et al., 2015)	67.11	52.19	45.04
WMD[†] w/o stopwords	72.02	70.05	42.41
DynaMax (Zhelezniak et al., 2019)	70.9	-	-
BERTScore[†] (Zhang et al., 2019)	67.26	50.93	44.77
WRD	*68.80*	*54.03*	*43.86*
WRD + **VC**(WR)	*74.23*	*66.82*	*49.35*
WRD + **VC**(SUP)	*77.03*	*72.66*	*55.90*
WRD + **VC**(SWC)	***77.92***	***74.43***	***56.70***
fastText – Additive Composition			
fastText[†]	67.85	60.95	51.42
fastText + WR[†] (Arora et al., 2017)	72.15	69.48	48.76
fastText + SUP[†] (Ethayarajh, 2018)	**76.22**	**74.24**	**53.70**
fastText – Considering Word Alignment			
WMD[†] (Kusner et al., 2015)	67.58	52.31	44.34
WMD[†] w/o stopwords	71.61	69.41	40.94
DynaMax (Zhelezniak et al., 2019)	76.6	-	-
BERTScore[†] (Zhang et al., 2019)	69.00	53.86	52.95
WRD	*73.31*	*62.10*	*56.70*
WRD + **VC**(WR)	*76.81*	*71.94*	*54.93*
WRD + **VC**(SUP)	*77.41*	***76.97***	*57.54*
WRD + **VC**(SWC)	***77.98***	*75.81*	***58.08***
Sent2Vec (Pagliardini et al., 2018)	-	75.5[*]	-
Skip-Thought[‡] (Kiros et al., 2015)	46	-	-
ELMo [‡] (Peters et al., 2018)	68	-	-

Table 5: **Pearson's r × 100** between the predicted scores and the gold scores. The best results in each dataset, word vector, and strategy for computing the textual similarity ("Additive composition" or "Considering Word Alignment") is in **bold**; and the best results regardless of the strategy are further **underlined**. Each row marked (†) is re-implemented by us. Each value marked (‡) is taken from Perone et al. (2018), and marked (∗) is taken from STS Wiki[6].

Pre-trained Word Vectors. Table 3a shows that using cosine similarity (i.e., ignoring the norm of the word vectors) yields a consistently higher correlation with human evaluations as opposed to using dot product or Euclidean distance (i.e., using the norm). This indicates that the angle of word vectors encodes the dissimilarity of words relatively well; on the other hand, the norm does not matter.

Converted Word Vectors. In light of the discussion in Section 6, we expect that the word dissimilarity of w and w' is "better" encoded in the angle between the converted word vectors $\langle \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{u}}' \rangle$ than in that between the pre-trained word vectors $\langle \boldsymbol{u}, \boldsymbol{u}' \rangle$. Table 3b demonstrates that the dissimilarity of words becomes increasingly accurately encoded by the angle of the word vectors as the conversion proceeds. This suggests that VC improves the word vectors in terms of the meaning of the words encoded in the direction vectors.

7.3 Ablation Study

We experimentally confirmed the effectiveness of the two proposed methods, WRD and VC, via the degree of performance improvement over the baseline, WMD. Table 4 shows the results. Following Kusner et al. (2015), we further experimented with stopword removal. In nearly all cases, WRD shows a higher predictive performance than WMD. We summarize some major findings as follows.

- As the word vectors are transformed by VC, the performance of WRD improves steadily. This is because WRD can directly utilize the weight and dissimilarity encoded in the norm and angle, whose quality is enhanced by VC. Conversely, WMD does not benefit from VC.
- WRD *without* stopwords removal achieves the best results. This is likely because WRD can softly compare differences in importance between stopwords using their norm.
- One might think that W (SIF weighting) can be directly used as the probability mass for the WMD computation because it is just a scaling factor for each word. "+ SIF weights" in Table 4 denotes such a computation. However, even when WMD removes stopwords and uses SIF directly, it does not reach the performance of WRD.

We obtained similar results for the other word vectors (see Appendix E for more details).

7.4 Benchmark Tasks

Finally, we compare the performance of the proposed methods (WRD and VC) with various base-

[6] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

line methods, including recent alignment-based methods such as **WMD** (Kusner et al., 2015), **BERTScore** (Zhang et al., 2019), and **DynaMax** (Zhelezniak et al., 2019). The results are shown in Table 5. We summarize some of our major findings as follows.

- Of the methods that consider word alignment, WRD + VC achieved the best performance. This is likely because other methods use Euclidean distance (Kusner et al., 2015) or dot product (Zhelezniak et al., 2019) as word similarity measures. These metrics cannot distinguish the two types of information (weight and dissimilarity) that each word vector holds separately by dividing it into a norm and a direction.
- BERTScore uses cosine similarity like WRD, but their scores were inferior to WRD on average. This can be attributed to the fact that BERTScore is completely disregarding the norm.
- Compared with strong baselines (WR, SUP), WRD using the same word vectors (**VC**(WR), **VC**(WR)) performed equally or better. This result is unexpected given that WR and SUP were originally proposed to create sentence vectors, and WRD simply uses them without tuning. Thus, we assume that considering word alignment is an inherently good hypothesis for STS.

See Appendix F for the comprehensive results using additional datasets and methods, including results semi-supervised and supervised approaches.

8 Connection to Other Methods

In this section, we present the relation between WRD, WMD, and ADD (the cosine similarity of additive composition; Equations 11 and 12), from the perspective of sentence representation.

Connection to Additive Composition. *ADD is a special case of WRD.* Indeed, given a discrete-distribution representation containing only one sentence vector, $\boldsymbol{\mu}_s^{\text{point}} = \{(\boldsymbol{s}, 1)\}$, the obvious EMD cost between them is equivalent to ADD.

$$\mathrm{EMD}(\boldsymbol{\mu}_s^{\text{point}}, \boldsymbol{\mu}_{s'}^{\text{point}}; c_{\cos}) = 1 - \cos(\boldsymbol{s}, \boldsymbol{s}') \quad (26)$$

The relationship between ADD and WRD becomes clearer when examining their sentence representations:

$$\boldsymbol{s} = \frac{1}{n}\sum_i \lambda_i \boldsymbol{u}_i, \quad \boldsymbol{\nu}_s = \frac{1}{Z}\sum_i \lambda_i \delta[\boldsymbol{u}_i], \quad (27\text{ a,b})$$

where $\delta[\cdot]$ is the Dirac delta function. As for the formulae, they are still quite similar. The key difference lies in the fact that ADD treats a sentence as a single vector (the barycenter of direction vectors), whereas WRD treats a sentence as a set of direction vectors. Thus, it is natural that WRD had a positive effect on STS tasks, given that STS tasks require the word alignment (i.e., they assume that words are treated disjointedly). On the other hand, ADD demonstrated higher performance on the topic similarity task[7]. For a task where it is sufficient to know the trend of the meaning of the whole sentence, it might be preferable to aggregate the meaning of the entire sentence into a single vector.

Connection to WMD. Why do WMD and WRD differ in performance on STS tasks even though both of them represent sentences as "bag-of-word-vectors" representations? Sentence representations for ADD and WMD are as follows:

$$\boldsymbol{s} = \frac{1}{n}\sum_i 1\cdot \boldsymbol{w}_i, \quad \boldsymbol{\mu}_s = \frac{1}{n}\sum_i 1\cdot \delta[\boldsymbol{w}_i]. \quad (28\text{ a,b})$$

Note that, barycenters (27a), (28a) for ADD are identical since $\lambda_i \boldsymbol{u}_i = \boldsymbol{w}_i$ holds by definition (10); on the other hand, the discrete distributions (27b) for WRD and that (28b) for WMD are quite different. WRD treats the norm λ as a weighting factor, as ADD implicitly does; in contrast, WMD assigns uniform weights to both long and short vectors. This is one reason why the most natural representation (28b) does not work well.

9 Conclusion

To solve the performance problem remaining for alignment-based STS methods, we proposed word rotator's distance (WRD), a new unsupervised, EMD-based STS metric. We first indicated that (i) the norm and angle of word vectors are good proxies for the importance of a word and the dissimilarity between words, respectively, and (ii) some previous methods "mix up" the norm and direction vectors. With this finding, WRD was designed so that the norm and angle of word vectors correspond to the probability mass and transportation cost in EMD, respectively. Moreover, we found that the latest powerful methods for sentence-vector estimation improve the norm and angle of word vectors (via vector converter; VC). In experiments on multiple STS benchmarks, the proposed methods outperformed not only alignment-based methods such as WMD but also powerful sentence vectors.

[7] SICK-R. See Appendix F for details.

Acknowledgments

We appreciate the helpful comments from the anonymous reviewers. We thank Emad Kebriaei for indicating how to normalize the sentence vectors. We also thank Benjamin Heinzerling, Masashi Yoshikawa, Hiroki Ouchi, Sosuke Kobayashi, Paul Reisert, Ana Brassard, and Shun Kiyono for constructive comments on the manuscript, and Masatoshi Suzuki and Goro Kobayashi for technical support. This work was supported by JSPS KAKENHI Grants JP19J21913 and JP19H04162.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *SemEval*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. In *SemEval*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 Task 1: Semantic Textual Similarity, Monolingual and Cross-Lingual Evaluation. In *SemEval*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity. In **SEM*, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic Textual Similarity. In **SEM*, pages 32–43.

David Alvarez-Melis, Tommi S. Jaakkola, and Stefanie Jegelka. 2018. Structured Optimal Transport. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 1771–1780.

Nikolay Arefyev, Pavel Ermolaev, and Alexander Panchenko. 2018. How much does a word weigh? Weighting word embeddings for word sense induction. *arXiv:1805.09209*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A Latent Variable Model Approach to PMI-based Word Embeddings. *TACL*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR*.

Steven Bird and Edward Loper. 2004. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *TACL*, 5:135–146.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional Semantics in Technicolor. In *ACL*, pages 136–145.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *SemEval*, pages 1–14.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, Yun Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *EMNLP (System Demonstrations)*, pages 169–174.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*, pages 2635–2639.

Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. 2019. Sentence Mover's Similarity: Automatic Evaluation for Multi-Sentence Texts. In *ACL*, pages 2748–2760.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In *EMNLP*, pages 670–680.

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NIPS*, pages 2292–2300.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, pages 4171–4186.

Kawin Ethayarajh. 2018. Unsupervised Random Walk Sentence Embeddings: A Strong but Simple Baseline. In *Rep4NLP*, pages 91–100.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation. *Computational Linguistics*, 41(4):665–695.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*, pages 873–882.

Mikhail Khodak, Nikunj Saunshi, Yingyu Liang, Tengyu Ma, Brandon Stewart, and Sanjeev Arora. 2018. A La Carte Embedding: Cheap but Effective Induction of Semantic Feature Vectors. In *ACL*, pages 12–22.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS*, pages 3294–3302.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From Word Embeddings To Document Distances. In *ICML*, volume 37, pages 957–966.

Tianlin Liu, Lyle Ungar, and João Sedoc. 2019a. Continual Learning for Sentence Representations Using Conceptors. In *NAACL*, pages 3274–3279.

Tianlin Liu, Lyle H. Ungar, and João Sedoc. 2019b. Unsupervised Post-processing of Word Vectors via Conceptor Negation. In *AAAI*, pages 6778–6785.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *CoNLL*, pages 104–113.

Xiaofei Ma, Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. 2019. Universal Text Representation from BERT: An Empirical Study. *arXiv:1910.07973*.

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 *SemEval*, pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *ICLR*.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features. In *NAACL*, pages 528–540.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*, pages 1532–1543.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv:1806.06259*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*, pages 2227–2237.

Gabriel Peyré and Marco Cuturi. 2019. Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A Word at a Time: Computing Word Relatedness Using Temporal Semantic Analysis. In *WWW*, pages 337–346.

Herbert Rubenstein and John B Goodenough. 1965. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633.

Filippo Santambrogio. 2015. *Optimal Transport for Applied Mathematicians*. Birkhäuser Basel.

Adriaan M J Schakel and Benjamin J Wilson. 2015. Measuring Word Significance using Distributed Representations of Words. *arXiv:1508.02297*.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning General Purpose Distributed Sentence Representations via Large Scale Multi-task Learning. In *ICLR*.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. DLS$@$CU: Sentence Similarity from Word Alignment. In *SemEval*, pages 241–246.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence Similarity from Word Alignment and Semantic Vector Composition. In *SemEval*, pages 148–153.

Vayer Titouan, Nicolas Courty, Romain Tavenard, Chapel Laetitia, and Rémi Flamary. 2019. Optimal Transport for structured data with application on graphs. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6275–6284.

Cédric Villani. 2009. *Optimal Transport*, 1 edition, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer Berlin Heidelberg.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *ICLR*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *TACL*, 3:345–358.

John Wieting, Taylor Berg-Kirkpatrick, Kevin Gimpel, and Graham Neubig. 2019. Beyond BLEU:Training Neural Machine Translation with Semantic Similarity. In *ACL*, pages 4344–4355.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. In *ACL*, pages 451–462.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv:1910.03771*.

Wei Xu, Chris Callison-Burch, and William B. Dolan. 2015. SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT). In *SemEval*, pages 1–11.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS*, pages 1–18.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. *arXiv:1904.09675*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance. In *EMNLP*, pages 563–578.

Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y Hammerla. 2019. Don't Settle for Average, Go for the Max: Fuzzy Sets and Max-Pooled Word Vectors. In *ICLR*.

A Resources Used in Experiments

A.1 Pre-trained Word Embeddings.

We used the following pre-trained word embeddings in our experiments.

- **GloVe** trained with Common Crawl (Pennington et al., 2014)[8]
- **word2vec** trained with Google News (Mikolov et al., 2013)[9]
- **fastText** trained with Common Crawl (Bojanowski et al., 2017)[10]
- **PSL**, the ParagramSL-999 embeddings, trained with the PPDB paraphrase database (Wieting et al., 2015)[11]
- **ParaNMT** trained with ParaNMT-50, a large scale English-English paraphrase database (Wieting and Gimpel, 2018)[12]
- **ELMo** pre-trained with 1 Billion Word Benchmark, a corpus with approximately 30 million sentences (Chelba et al., 2014) (BiLSTM hidden size of 4096, output size of 512, and 2 highway layers) (Peters et al., 2018)[13]
- **BERT-Large** pre-trained with the BooksCorpus (800M words) and English Wikipedia (2500M words) (uncased, 24 layers, hidden size of 1024, 16 self-attention heads, and 340M parameters) (Devlin et al., 2019)[14]. We use the PyTorch implementation of BERT (Wolf et al., 2019)[15].

A.2 Word Similarity Datasets

We used the following nine different datasets for the word similarity task.

- **MEN** (Bruni et al., 2012)
- **MTurk287** (Radinsky et al., 2011)
- **MC30** (Miller and Charles, 1991)
- **RW** (Luong et al., 2013)
- **RG65** (Rubenstein and Goodenough, 1965)
- **SCWS** (Huang et al., 2012)
- **SimLex999** (Hill et al., 2015)
- **WS353** (Finkelstein et al., 2002)

A.3 STS Datasets Used in Experiments

We used the following STS datasets in our experiments.

- **STS'12** (Agirre et al., 2012), **STS'13** (Agirre et al., 2013), **STS'14** (Agirre et al., 2014), and **STS'15** (Agirre et al., 2015): semantic textual similarity shared tasks in SemEval
- **STS-B**: semantic textual similarity benchmark (Cer et al., 2017), which is the collection from SemEval STS tasks 2012–2017 (Agirre et al., 2012, 2013, 2014, 2015, 2016; Cer et al., 2017)
- **Twitter**: paraphrase and semantic similarity in twitter (PIT) task in SemEval 2015 (Xu et al., 2015)
- **SICK-R**: SemEval 2014 semantic relatedness task (Marelli et al., 2014)

Tokenization. In every experiment, we first tokenized all the STS datasets other than the Twitter dataset with a modified NLTK (Bird and Loper, 2004; Ethayarajh, 2018)[16]. The Twitter dataset has already been tokenized by the organizer. We then lowercased all corpora to conduct experiments under the same conditions with cased embeddings and non-cased embeddings.

B Contextualized Word Embeddings on Unsupervised STS

BERT (Devlin et al., 2019) and its variants have not yet shown good results on *unsupervised* STS (note that, in a supervised or semi-supervised setting where there exists training data or external resources, BERT-based models show the current, best results). One particularly promising usage of BERT-based models for unsupervised STS is BERTScore (Zhang et al., 2019), which was originally proposed as an automatic evaluation metric. However, our preliminary experiments[17] show that BERTScore performs poorly on unsupervised STS. Since BERTScore is definitely promising as a method, we reported the results using non-contextualized vectors, e.g., GloVe, where we confirmed a higher performance compared to BERT. Needless to say, the application of BERT-based models to unsupervised STS is a very important future research topic.

[8] https://nlp.stanford.edu/projects/glove/

[9] https://code.google.com/archive/p/word2vec/

[10] https://fasttext.cc/docs/en/english-vectors.html

[11] http://www.cs.cmu.edu/~jwieting/

[12] https://github.com/kawine/usif

[13] https://allennlp.org/elmo

[14] https://github.com/google-research/bert

[15] https://github.com/huggingface/transformers

[16] https://github.com/kawine/usif

[17] We used BERT-large and RoBERTa-large. For the embedding, we used either the last layer or the concatenation of all the layers. In the original paper, which allows the use of teacher data, the development set was used to select the layer.

Algorithm 1 Word Rotator's Distance (WRD)

```
Input: a pair of sentences $s = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_n)$, $s' = (\boldsymbol{w}'_1, \dots, \boldsymbol{w}'_{n'})$
 1: $Z \leftarrow \sum_{i=1}^{n} \|\boldsymbol{w}_i\| \in \mathbb{R}$
 2: $Z' \leftarrow \sum_{j=1}^{n'} \|\boldsymbol{w}'_j\| \in \mathbb{R}$
 3: $\boldsymbol{m}_s \leftarrow \frac{1}{Z}(\|\boldsymbol{w}_1\|, \dots, \|\boldsymbol{w}_n\|) \in \mathbb{R}^n$
 4: $\boldsymbol{m}_{s'} \leftarrow \frac{1}{Z'}(\|\boldsymbol{w}'_1\|, \dots, \|\boldsymbol{w}'_{n'}\|) \in \mathbb{R}^{n'}$
 5: for $i \leftarrow 1$ to $n$ do
 6:     for $j \leftarrow 1$ to $n'$ do
 7:         $\boldsymbol{C}_{ij} \leftarrow 1 - \cos(\boldsymbol{w}_i, \boldsymbol{w}'_j)$
 8:     end for
 9: end for
10: $\mathrm{WRD}(s, s') \leftarrow \mathrm{EMD}(\boldsymbol{m}_s, \boldsymbol{m}_{s'}; \boldsymbol{C})$
Output: $\mathrm{WRD}(s, s') \in \mathbb{R}$
```

C Algorithm of Word Rotator's Distance

The algorithm used in the actual computation of WRD is shown in Algorithm 1.

For EMD computation, off-the-shelf libraries can be used[18]. Note that most EMD (optimal transport) libraries take two probabilities (mass) $\boldsymbol{m} \in \mathbb{R}^n$, $\boldsymbol{m}' \in \mathbb{R}^{n'}$ and a cost matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n'}$ with $\boldsymbol{C}_{ij} = d(\boldsymbol{x}_i, \boldsymbol{x}'_j)$ as inputs. Parameters ($\boldsymbol{m}$, $\boldsymbol{m}'$, $\boldsymbol{C}$) have the same information as ($\boldsymbol{\mu}$, $\boldsymbol{\mu}'$, d), introduced in Section 4.3. The notation of Algorithm 1 follows this style.

The cosine distance $1 - \cos(\boldsymbol{w}_i, \boldsymbol{w}'_j)$ in line 7 of Algorithm 1 is equivalent to $1 - \cos(\boldsymbol{u}_i, \boldsymbol{u}'_j)$ in Equation 18. We adopted the former simply to reduce the computation steps.

D Algorithms of Vector Converter

Algorithm 2 summarizes the overall procedure of word vector converter f_{VC} (22).

When computing Algorithm 2, we set hyperparameters as $D_{\mathrm{A}} = 3$, $\alpha_{\mathrm{N}} = 2$, and $a_{\mathrm{W}} = 10^{-3}$, following Mu and Viswanath (2018), Liu et al. (2019a), and Arora et al. (2017), respectively, without tuning. We used the unigram probability $\mathbb{P}$ of English words estimated using the enwiki dataset, preprocessed by Arora et al. (2017)[19].

See Table 6 for an overview of the existing methods. There are many possible combinations of f_1, f_2, and f_3, and exploring them is a good direction for future work.

Algorithm 2 Word Vector Converter (VC), induced from **A**ll-but-the-top (Mu and Viswanath, 2018) or conceptor **N**egation (Liu et al., 2019a), SIF **W**eighting (Arora et al., 2017), and common component **R**emoval (Arora et al., 2017).

```
Input: pre-trained word vectors $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_{|\mathcal{V}|}\} \subseteq \mathbb{R}^d$, sentences in interest $\mathcal{S} = \{s_1, \dots, s_{|\mathcal{S}|}\}$, word unigram probability $\mathbb{P}\colon \mathcal{V} \to [0, 1]$, and constants $D_{\mathrm{A}}$ (or $\alpha_{\mathrm{N}}$), $a_{\mathrm{W}}$
    Compute parameters of $f_1$:
    ··· if using All-but-the-top:
    $\overline{\boldsymbol{w}} \leftarrow \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{w}_i \in \mathbb{R}^D$
    for $i \leftarrow 1$ to $|\mathcal{V}|$ do
        $\overline{\boldsymbol{w}}_i \leftarrow \boldsymbol{w}_i - \overline{\boldsymbol{w}}$
    end for
    $\boldsymbol{u}_1, \dots, \boldsymbol{u}_{D_{\mathrm{A}}} \leftarrow \mathrm{PCA}(\{\overline{\boldsymbol{w}}_1, \dots, \overline{\boldsymbol{w}}_{|\mathcal{V}|}\})$
                                        ▷ top $D_{\mathrm{A}}$ singular vectors
    $\boldsymbol{A}_1 \leftarrow \boldsymbol{I} - \sum_{j=1}^{D_{\mathrm{A}}} \boldsymbol{u}_j \boldsymbol{u}_j^\top \in \mathbb{R}^{D \times D}$
    $\boldsymbol{b}_1 \leftarrow \overline{\boldsymbol{w}} \in \mathbb{R}^D$
    ··· else if using conceptor Negation:
    $\boldsymbol{R} \leftarrow \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \boldsymbol{w}_i \boldsymbol{w}_i^\top \in \mathbb{R}^{D \times D}$
    $\boldsymbol{C} \leftarrow \boldsymbol{R}(\boldsymbol{R} + \alpha_{\mathrm{N}}{}^{-2} \boldsymbol{I})^{-1} \in \mathbb{R}^{D \times D}$
    $\boldsymbol{A}_1 \leftarrow \boldsymbol{I} - \boldsymbol{C} \in \mathbb{R}^{D \times D}$
    $\boldsymbol{b}_1 \leftarrow \boldsymbol{0} \in \mathbb{R}^D$
    Compute parameters of $f_3$:
    for $i \leftarrow 1$ to $|\mathcal{S}|$ do
        $\boldsymbol{s}_i \leftarrow \frac{1}{|s_i|} \sum_{w \in s_i} \alpha_2(w) \boldsymbol{A}_1(\boldsymbol{w} - \boldsymbol{b}_1)$
    end for
    $\boldsymbol{v} \leftarrow \mathrm{PCA}(\{\boldsymbol{s}, \dots, \boldsymbol{s}_{|\mathcal{S}|}\})$
                                        ▷ first singular vector
    $\boldsymbol{A}_3 \leftarrow \boldsymbol{I} - \boldsymbol{v}\boldsymbol{v}^\top \in \mathbb{R}^{d \times d}$
    Convert word vectors:
    for $i \leftarrow 1$ to $|\mathcal{V}|$ do
        $\alpha_2(w) \leftarrow a_{\mathrm{W}} / (\mathbb{P}(w) + a_{\mathrm{W}})$
        $\widetilde{\boldsymbol{w}}_i \leftarrow \boldsymbol{A}_3(\alpha_2(w) \boldsymbol{A}_1(\boldsymbol{w}_i - \boldsymbol{b}_1))$
    end for
Output: Converted word vectors $\{\widetilde{\boldsymbol{w}}_1, \cdots, \widetilde{\boldsymbol{w}}_{|\mathcal{V}|}\}$
```

E Full Results of Ablation Study

See Table 7 for full results.

F Full Results of Comparative Experiments

See Table 8 for full results in an unsupervised settings. See Table 9 for full results in an semisupervised and supervised settings.

[18] In our experiments, we used the well-developed python optimal transport (POT) library: https://github.com/rflamary/POT/. In particular, `ot.emd2()` was used.

[19] https://github.com/PrincetonML/SIF/

[20] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

	f_1	α_2	f_3
	denoising word vectors	scaling	denoisng sentence vectors
well-known heuristic	–	Stop Words Removal	–
well-known heuristic	–	IDF (Inverse Document Frequency)	–
Arora et al. (2017)	–	SIF (Smoothed Inverse Frequency)	Common Component Removal
Mu and Viswanath (2018)	all-but-the-top	–	–
Ethayarajh (2018)	Dimension-wise Normalization	uSIF (Unsupervised SIF)	Piecewise Common Component Removal
Liu et al. (2019b)	Conceptor Negation	–	–
Liu et al. (2019a)	–	SIF	Conceptor Removal

Table 6: Unsupervised sentence encoders.

	WMD	**WRD**	WMD	**WRD**
Removing Stopwords			✓	✓
GloVe	62.56	**64.66**	**71.34**	**71.13**
GloVe + A	65.74	**68.83**	**75.19**	**75.19**
GloVe + AW	63.34	**77.21**	74.41	**76.44**
GloVe + A + SIF weights	76.81	-	76.56	-
GloVe + **VC**(AWR)	61.42	**79.20**	72.81	**78.60**
word2vec	67.26	**71.05**	72.41	**73.19**
word2vec + A	67.22	**71.32**	72.46	**73.65**
word2vec + AW	63.89	**71.59**	71.59	**74.91**
word2vec + A + SIF weights	74.70	-	73.98	-
word2vec + **VC**(AWR)	62.76	**77.07**	70.22	**76.43**
fastText	61.64	**67.93**	70.46	**74.07**
fastText + A	64.00	**69.95**	73.52	**76.45**
fastText + AW	61.15	**78.26**	72.63	**77.64**
fastText + A + SIF weights	75.50	-	75.06	-
fastText + **VC**(AWR)	59.78	**79.14**	71.27	**78.62**

Table 7: **Pearson's $r \times 100$** between the predicted scores and the gold scores for each word vector (each row) and each method (each column). The STS-B dataset (dev) is used. The best result and results where the difference from the best < 0.5 in each row are in **bold**, and the best result in each word vector is further **underlined**.

[21] http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark

	STS'12	STS'13	STS'14	STS'15	STS-B	Twitter	SICK-R
GloVe – Additive Composition							
GloVe†	53.04	45.52	57.97	56.08	45.57	29.35	66.79
GloVe + WR (Arora et al., 2017)	56.2	56.6	68.5	71.7	-	48.0	72.2
GloVe + WR† (Arora et al., 2017)	60.57	54.99	67.74	67.74	62.85	40.03	69.32
GloVe + SUP (Ethayarajh, 2018)	**64.9**	**63.6**	**74.4**	**76.1**	**71.5**	-	**73.0**
GloVe + SUP† (Ethayarajh, 2018)	64.85	62.50	73.69	74.38	71.03	**50.24**	72.34
GloVe – Considering Word Alignment							
WMD GloVe† (Kusner et al., 2015)	55.74	44.18	60.24	67.11	52.19	45.04	61.91
WMD GloVe w/o stopwords† (Kusner et al., 2015)	60.67	53.45	67.63	72.02	70.05	42.41	63.31
DynaMax GloVe (Zhelezniak et al., 2019)	58.2	53.9	65.1	70.9	-	-	-
BERTScore GloVe† (Zhang et al., 2019)	52.81	47.23	62.06	67.26	50.93	44.77	65.28
WRD GloVe	*58.28*	*48.79*	*62.31*	*68.80*	*54.03*	*43.86*	*63.84*
WRD GloVe + **VC**(WR)	*62.96*	*56.88*	*68.73*	*74.23*	*66.82*	*49.35*	*66.94*
WRD GloVe + **VC**(SUP)	*64.28*	***58.19***	*71.10*	*77.03*	*72.66*	*55.90*	*67.29*
WRD GloVe + **VC**(SWC)	***64.61***	*58.00*	***72.20***	***77.92***	*74.43*	*56.70*	***67.51***
WRD GloVe + **VC**(SUC)	*64.39*	*57.70*	*71.87*	*77.63*	***74.96***	***57.27***	*65.82*
word2vec – Additive Composition							
word2vec†	61.67	53.07	67.63	67.45	61.54	30.54	**72.51**
word2vec + WR† (Arora et al., 2017)	62.79	**58.55**	71.11	70.41	67.49	**35.59**	70.78
word2vec + SUP† (Ethayarajh, 2018)	**63.27**	58.50	**71.72**	**72.97**	**69.39**	34.72	70.51
word2vec – Considering Word Alignment							
WMD word2vec† (Kusner et al., 2015)	55.89	44.52	60.24	66.46	56.10	**64.05**	39.53
WMD word2vec w/o stopwords† (Kusner et al., 2015)	58.14	49.95	65.22	70.54	67.46	36.00	62.41
DynaMax word2vec (Zhelezniak et al., 2019)	53.7	**59.5**	68.0	74.2	-	-	-
BERTScore word2vec† (Zhang et al., 2019)	47.83	43.54	56.26	62.06	49.16	34.07	58.75
WRD word2vec	*59.14*	*51.41*	*65.36*	*72.39*	*72.39*	*41.44*	*66.31*
WRD word2vec + **VC**(WR)	*61.45*	*55.98*	*68.52*	*74.86*	*70.13*	*43.42*	***66.76***
WRD word2vec + **VC**(SUP)	*61.85*	*55.38*	*68.96*	*75.30*	*71.19*	*42.86*	*66.11*
WRD word2vec + **VC**(SWC)	***62.01***	*55.52*	***69.46***	***75.72***	*72.45*	*44.41*	*65.91*
WRD word2vec + **VC**(SUC)	*61.37*	*55.04*	*68.85*	*75.23*	***72.54***	*45.16*	*64.14*
fastText – Additive Composition							
fastText†	59.76	52.79	67.42	67.85	60.95	51.42	70.44
fastText + WR† (Arora et al., 2017)	64.03	59.90	72.88	72.15	69.48	48.76	**72.19**
fastText + SUP† (Ethayarajh, 2018)	**64.39**	**62.33**	**74.82**	**76.22**	**74.24**	**53.70**	72.13
fastText – Considering Word Alignment							
WMD fastText† (Kusner et al., 2015)	55.27	44.39	60.09	67.58	52.31	44.34	62.21
WMD fastText w/o stopwords† (Kusner et al., 2015)	60.00	52.29	66.87	71.61	69.41	40.94	62.84
DynaMax fastText (Zhelezniak et al., 2019)	60.9	**60.3**	69.5	76.6	-	-	-
BERTScore fastText† (Zhang et al., 2019)	51.95	45.86	61.66	69.00	53.86	52.95	64.69
WRD fastText	*58.84*	*50.74*	*64.60*	*73.31*	*62.10*	*56.70*	*64.90*
WRD fastText + **VC**(WR)	*63.50*	*58.44*	*70.26*	*76.81*	*71.94*	*54.93*	***67.85***
WRD fastText + **VC**(SUP)	***64.22***	*58.84*	*71.41*	*77.41*	***76.97***	*57.54*	*67.36*
WRD fastText + **VC**(SWC)	*64.17*	*58.92*	***72.03***	***77.98***	*75.81*	*58.08*	*67.39*
WRD fastText + **VC**(SUC)	*63.76*	*58.60*	*71.46*	*77.48*	*75.84*	***58.19***	*65.56*
Sent2Vec (Pagliardini et al., 2018)	-	-	-	-	75.5*	-	-
Skip-Thought‡ (Kiros et al., 2015)	41	29	40	46	-	-	-
ELMo (All layers, 5.5B)‡ (Peters et al., 2018)	55	53	63	68	-	-	-

Table 8: **Pearson's $r \times 100$** between the predicted scores and the gold scores for each method (each row) and each dataset (each column). The best results in each dataset, word vector, and strategy for computing textual similarity ("Additive composition" or "Considering Word Alignment") is in **bold**; and the best results regardless of the strategy for computing textual similarity are further **underlined**. The results of our methods are *slanted*. Each row marked (†) is re-implemented by us. Each value marked (‡) is taken from Perone et al. (2018). Each value marked (∗) is taken from STS Wiki[20].

	STS'12	STS'13	STS'14	STS'15	STS-B	Twitter	SICK-R
Semi-supervised							
PPDB supervision – Additive Composition							
PSL† (Wieting et al., 2016)	55.07	48.00	61.63	61.21	51.32	36.29	66.52
PSL + WR (Arora et al., 2017)	59.5	61.8	73.5	76.3	72.0*	49.0	**72.9**
PSL + WR† (Arora et al., 2017)	64.76	62.34	73.77	73.82	70.73	45.97	70.88
PSL + UP (Ethayarajh, 2018)	**65.8**	**65.2**	**75.9**	**77.6**	**74.8**	-	72.3
PSL + UP† (Ethayarajh, 2018)	65.79	64.48	75.70	76.79	74.13	**50.64**	71.80
PPDB supervision – Considering Word Alignment							
WMD PSL† (Kusner et al., 2015)	55.52	44.52	61.39	69.38	56.93	50.57	61.78
WMD PSL w/o stopwords† (Kusner et al., 2015)	61.28	54.13	69.45	74.14	70.93	46.31	63.24
DynaMax PSL (Zhelezniak et al., 2019)	58.2	54.3	66.2	72.4	-	-	-
BERTScore PSL† (Zhang et al., 2019)	56.90	51.31	66.39	71.85	60.33	49.47	67.40
WRD PSL	*57.84*	*48.84*	*63.41*	*71.20*	*59.03*	*48.60*	*64.29*
WRD PSL + **VC**(WR)	*65.13*	*60.07*	*71.29*	*77.20*	*72.71*	*52.02*	*67.44*
WRD PSL + **VC**(SUP)	***65.60***	***60.24***	***72.51***	***77.61***	***74.31***	***54.02***	***67.72***
ParaNMT supervision – Additive Composition							
ParaNMT† (Wieting and Gimpel, 2018)	67.77	62.35	77.29	**79.51**	**79.85**	**49.53**	**74.80**
ParaNMT + WR† (Arora et al., 2017)	67.81	64.62	77.00	77.87	79.74	39.42	73.48
ParaNMT + UP (Ethayarajh, 2018)	68.3	**66.1**	**78.4**	79.0	79.5	-	73.5
ParaNMT + UP† (Ethayarajh, 2018)	**68.47**	65.29	78.29	78.95	79.43	46.67	73.28
ParaNMT supervision – Considering Word Alignment							
WMD ParaNMT† (Kusner et al., 2015)	60.06	47.00	64.01	70.40	56.43	46.95	65.06
WMD ParaNMT w/o stopwords† (Kusner et al., 2015)	63.02	54.39	70.70	73.88	72.65	47.14	64.80
DynaMax ParaNMT (Zhelezniak et al., 2019)	66.0	**65.7**	**75.9**	**80.1**	-	-	-
BERTScore ParaNMT† (Zhang et al., 2019)	57.41	49.35	65.88	71.66	61.24	**55.44**	67.23
WRD ParaNMT	*65.89*	*56.05*	*72.03*	*78.01*	*74.12*	*53.83*	*69.49*
WRD ParaNMT + **VC**(WR)	***67.95***	*61.94*	*75.70*	*79.96*	*79.01*	*50.19*	***70.42***
WRD ParaNMT + **VC**(SUP)	*67.68*	*61.98*	*75.57*	*79.94*	***79.06***	*52.44*	*69.70*
SNLI supervision							
USE (Transformer)‡ (Cer et al., 2018)	61	64	71	74	-	-	-
InferSent‡ (Conneau et al., 2017)	61	56	68	71	75.8*	-	-
GenSen (+STN +Fr +De +NLI +2L +STP) (Subramanian et al., 2018)	-	-	-	-	79.2	-	88.8
Supervised							
XLNet-large (ensemble) (Yang et al., 2019)	-	-	-	-	93.0	-	-

Table 9: **Pearson's $r \times 100$** between the predicted scores and the gold scores for each method (each row) and each dataset (each column). The best results in each dataset, word vector, and strategy for computing textual similarity ("Additive composition" or "Considering Word Alignment") is in **bold**; and the best results regardless of the strategy for computing textual similarity are further **underlined**. The results of our methods are *slanted*. Each row marked (†) is re-implemented by us. Each value marked (‡) is taken from Perone et al. (2018). Each value marked ($*$) is taken from STS Wiki[21].