# 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

A key principle in assessing textual similarity is measuring the degree of semantic overlap between two texts by considering the word alignment. Such alignment-based approaches are intuitive and interpretable; however, they are empirically inferior to the simple cosine similarity between general-purpose sentence vectors. To address this issue, we focus on and demonstrate the fact that the *norm* of word vectors is a good proxy for word importance, and their *angle* is a good proxy for word similarity. Alignment-based approaches do not distinguish them, whereas sentence-vector approaches automatically use the norm as the word importance. Accordingly, we propose a method that first decouples word vectors into their norm and direction, and then computes alignment-based similarity using earth mover's distance (i.e., optimal transport cost), which we refer to as *word rotator's distance.* Besides, we find how to "grow" the norm and direction of word vectors (*vector converter*), which is a new systematic approach derived from sentence-vector estimation methods. On several textual similarity datasets, the combination of these simple proposed methods outperformed not only alignment-based approaches but also strong baselines. [1]

## 1 Introduction

This paper addresses the semantic textual similarity (STS) task, 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 the loss functions and automatic evaluation metrics of text generation tasks because a requirement of these metrics is precisely the calculation of STS (Wieting et al., 2019; Zhao et al., 2019; Zhang et al., 2019).

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

There are two major approaches to tackling STS. One is to measure the degree of semantic overlap between 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 approach involves generating general-purpose sentence vectors from two texts (typically comprising word vectors), and then calculating their similarity, which we refer to as *sentence-vector approaches* (Arora et al., 2017; Ethayarajh, 2018). Alignment-based approaches are consistent with human intuition about 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. In contrast, sentence-vector approaches automatically exploit the norm of the word vectors as the relative importance of words.

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

Our primary contributions are as follows.

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



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2944–2960,
November 16–20, 2020. 

the norm and direction of word vectors.

- To enhance the proposed WRD, we utilize a new word-vector conversion mechanism, which is formally induced from 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* English 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). Note that *semi-supervised* methods that utilize such external corpora have also 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 $\mathcal{V}$

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

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

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

## 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 utilizing deep sentence encoders (Kiros et al., 2015; Conneau et al., 2017; Cer et al., 2018), learning and using word vectors optimized for summation (Pagliardini et al., 2018; Wieting and Gimpel, 2018), and 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 utilizing 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 Issues

### 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

[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.

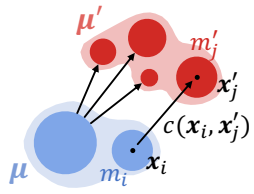


Figure 1:
Earth Mover's Distance.

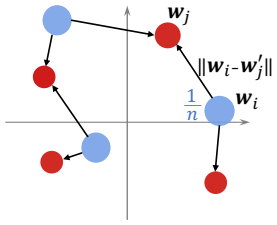


Figure 2:
Word Mover's Distance.

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

1. Two **probability distributions**, $\boldsymbol{\mu}$ (initial arrangement) and $\boldsymbol{\mu}'$ (final arrangement)[3]:

$$\boldsymbol{\mu} = \left\{(\boldsymbol{x}_i,\, m_i)\right\}_{i=1}^{n},\ \boldsymbol{\mu}' = \left\{(\boldsymbol{x}'_j,\, m'_j)\right\}_{j=1}^{n'}. \tag{3}$$

   Here, $\boldsymbol{\mu}$ denotes a probability distribution, where each point $\boldsymbol{x}_i \in \mathbb{R}^d$ has a probability mass $m_i \in [0, 1]$ ($\sum_i m_i = 1$). In Figure 1, each circle represents a pair $(\boldsymbol{x}_i, m_i)$, where the location and size of the circle represent a vector $\boldsymbol{x}_i$ 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, $c(\boldsymbol{x}_i, \boldsymbol{x}'_j)$ determines the transportation cost per unit amount (distance) between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}'_j$.

The EMD between $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$ is then defined via the following optimization problem:

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

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

Here, a solution $\boldsymbol{T} \in \mathbb{R}_{\geq 0}^{n \times n'}$ denotes a transportation plan, where each element $\boldsymbol{T}_{ij}$ represents the mass transported from $\boldsymbol{x}_i$ to $\boldsymbol{x}'_j$. In summary, $\mathrm{EMD}(\boldsymbol{\mu}, \boldsymbol{\mu}'; c)$ is the cost of the best transportation plan between two distributions $\boldsymbol{\mu}$ and $\boldsymbol{\mu}'$.

**Side Benefit: Alignment.** Under the above optimization, if the locations $\boldsymbol{x}_i$ and $\boldsymbol{x}'_j$ are *close* (i.e., if the transportation cost $c(\boldsymbol{x}_i, \boldsymbol{x}'_j)$ is small), they are likely to be *aligned* (i.e., $\boldsymbol{T}_{ij}$ may be assigned a large value). In this way, EMD can be considered to align the points of two discrete distributions. This is one reason we adopt EMD as a key technology in the computation of STS.

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

### 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) field. This study is strongly inspired by this work. We introduce WMD prior to presenting the proposed method.

WMD is the cost of transporting a set of word vectors in an embedding space (Euclidean space) (Figure 2). Formally, after removing stopwords, Kusner et al. (2015) regard each sentence $s$ as a uniformly weighted distribution $\boldsymbol{\mu}_s$ comprising word vectors (i.e., bag-of-word-vectors distribution):

$$\boldsymbol{\mu}_s := \left\{(\boldsymbol{w}_i,\, \frac{1}{n})\right\}_{i=1}^{n},\ \boldsymbol{\mu}_{s'} := \left\{(\boldsymbol{w}'_j,\, \frac{1}{n'})\right\}_{j=1}^{n'}. \tag{7}$$

In Figure 2, each circle represents each word, where the location and size of the circle represent the position vector $\boldsymbol{w}_i$ and its weight $\frac{1}{n}$, respectively. Next, Euclidean distance is used as the transportation cost between word vectors

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

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

$$\mathrm{WMD}(s, s') := \mathrm{EMD}(\boldsymbol{\mu}_s, \boldsymbol{\mu}_{s'}; c_{\mathrm{E}}). \tag{9}$$

### 4.3 Issues with Word Mover's Distance

Despite its intuitive formulation, the WMD often misaligns words with each other, and the STS performance of WMD is less than that of recent methods. For example, by WMD, "noodle" and "snack" may be aligned rather than "noodle" and "pho" (a type of Vietnamese noodle).

## 5 Word Rotator's Distance

Here, we first discuss the roles of the norm and direction of word vectors. Then, we describe issues with WMD from the perspective of the roles of the norm and direction. Finally, we present the proposed method, i.e., word rotator's distance, which can resolve the issues with WMD.

### 5.1 Roles of Norm and Direction

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

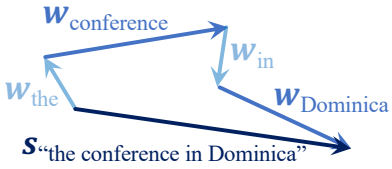


Figure 3: The addition operation implicitly uses the norm of the vectors as the weighting factor.

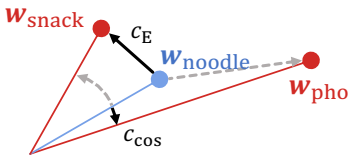


Figure 4: Euclidean distance "mixes up" the norm (a weighting factor for each word) and direction vectors (for word dissimilarity).

- **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 (i.e., the difference between the direction of these vectors) approximates the (dis)similarity of two words.

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

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

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 used in STS tasks and many downstream tasks (Mitchell and Lapata, 2010; Mikolov et al., 2013; Wieting et al., 2016).

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

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

**Norm as Weighting Factor.** Equation 11 may initially appear to treat each word vector equally. However, several previous studies have confirmed that the norm of word vectors has large dispersion (Schakel and Wilson, 2015; Arefyev et al., 2018). In other words, a sentence vector would contain word vectors of various lengths. In such cases, a word vector with a large norm will dominate in the resulting sentence vector, and vice versa (Figure 3). Here the usefulness of additive composition (i.e., 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.

In addition, the following are known about the relationship between word vector norm and the word importance: (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 of these observations suggest that the norm serves as a weighting factor in cases where additive composition is effective.

**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

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

It is widely known that the cosine similarity of word vectors trained on the basis of the distributional hypothesis well approximates word similarity (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 confirmed 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 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

[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 | **3.39** | 4.62 |
| *pho* | **3.52** | - | 4.52 | 5.60 |
| *snack* | **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 (closest word) is shown in **bold**. Inappropriate alignments by Euclidean distance are **underlined**. Here, pre-trained word2vec (Mikolov et al., 2013) was used.

words can be measured by angle, WMD uses Euclidean distance, which *mixes* the weighting factor and dissimilarity.

The problematic nature of this *mixing* can be explained as follows. Euclidean transportation cost (8) would misestimate the similarity of word pairs as low$_{\langle A \rangle}$, whose meanings are close$_{\langle B \rangle}$ but whose concreteness or importance is very different$_{\langle C \rangle}$, e.g., "noodle" and "pho" (Figure 4). This is clear from the relationship between the Euclidean (8) and cosine distances (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 large$_{\langle A \rangle}$ even if $c_{\cos}(\boldsymbol{w}, \boldsymbol{w}')$ is small$_{\langle B \rangle}$, as long as $|\lambda - \lambda'|$ is large$_{\langle C \rangle}$. Note that 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 (i.e., 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. The proposed method considers each sentence as a discrete distribution on the unit hypersphere and calculates EMD on this hypersphere (Figure 5). Here, the alignment of the direction vectors corresponds to a rotation on the unit hypersphere; thus, we refer to the proposed method as *word rotator's distance (WRD)*.

Formally, we consider each sentence $s$ as a discrete distribution $\boldsymbol{\nu}_s$ comprising direction vectors

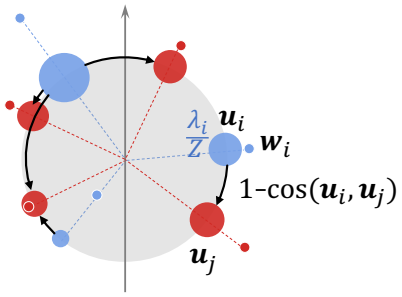


Figure 5: Word Rotator's Distance.

weighted by their norm (bag-of-direction-vectors distribution)

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

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 $\lambda_i / Z$, 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, a rotation cost is required to align words. Then, the WRD between two sentences is given as

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

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

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

**Algorithm.** To ensure reproducibility, we show the specific (and fairly simple) algorithm and implementation guidelines for WRD in Appendix C.

## 6 Vector Converter-enhanced WRD

To further improve the performance of WRD, we attempted to integrate existing methods to estimate latent sentence vectors, i.e., the most powerful sentence encoders for STS, into WRD. However, determining a method to combine sentence-vector estimation methods with WRD is not straightforward task 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., 2019b,a), and have achieved success in many NLP applications, including STS. Given pre-trained vectors of words comprising a sentence, these methods allow us to estimate the latent sentence vectors that generated the word vectors. Such methods can be summarized in the following form

$$\mathrm{Encode}(s) = f_3\left(\frac{1}{n}\sum_{w\in s}\alpha_2(w)f_1(\boldsymbol{w})\right), \quad (20)$$

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;
- $f_3\colon \mathbb{R}^D \to \mathbb{R}^D$ "denoises" each sentence vector.

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

**Word Vector Converter.** Note that all of the existing denoising function $f_3$ is linear; thus, Equation 20 can be rewritten as

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

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

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

### 6.2 Norm and Direction

We believe that the vector converter improves the norm and direction of pre-trained word vectors.

**Norm as Weighting Factor.** In Section 5, given the success of additive composition, we proposed the use of norms to weight words. In addition, in Section 6.1, we confirmed that the sentence-vector estimation methods, which have achieved greater success in STS than standard additive composition, simply sum the transformed word vectors (i.e., 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.** On the bases of the random-walk LM (Arora et al., 2016), the denoising function $f_1$ 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 word dissimilarity (Mu and Viswanath, 2018). Further, the functions $\alpha_2$ and $f_3$ assume a more realistic LM (Arora et al., 2017). Thus, VC is expected to further improve the isotropy of the vector space and make the angle of the word vector a better proxy for word dissimilarity.

### 6.3 Vector Converter-enhanced WRD

As we discussed previously, converted word vectors $\{\widetilde{\boldsymbol{w}}\}$ may have preferable properties in terms of their norm and direction, and they remain word vectors (i.e., they are no longer sentence vectors); thus, $\{\widetilde{\boldsymbol{w}}\}$ can be used as is for the input of WRD. Let $\widetilde{\lambda}$ and $\widetilde{\boldsymbol{u}}$ denote the norm and direction vector of $\widetilde{\boldsymbol{w}}$, resp.; then, a variant of WRD using $\{\widetilde{\boldsymbol{w}}\}$ is

$$\widetilde{\boldsymbol{\nu}}_s := \left\{(\widetilde{\boldsymbol{u}}_i, \frac{\widetilde{\lambda}_i}{Z})\right\}_{i=1}^{n}, \widetilde{\boldsymbol{\nu}}_{s'} := \left\{(\widetilde{\boldsymbol{u}}'_j, \frac{\widetilde{\lambda}_j}{Z'})\right\}_{j=1}^{n'}, \quad (23)$$

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

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

## 7 Experiments

In this section, we experimentally confirm our hypotheses about the norm and direction vectors and the performance of WRD and VC.

For word vectors, we used the standard **GloVe** (Pennington et al., 2014), **word2vec** (Mikolov et al., 2013), and **fastText** (Bojanowski et al., 2017). We did not use BERT (Devlin et al., 2019) and its variants because they are currently ineffective in unsupervised STS tasks. Refer to the Appendix B for additional details).

For STS datasets, we used **STS'15** (Agirre et al., 2015) for comparison with Zhelezniak et al. (2019); Kiros et al. (2015); Peters et al. (2018), STS benchmark (**STS-B**) (Cer et al., 2017), one of the most actively used datasets, and **Twitter'15** (Xu et al., 2015) to validate methods against casual writing.

For VC, we used the followings algorithms.

- $f_1$: **A**ll-but-the-top (Mu and Viswanath, 2018), sentence-wise feature **S**caling(Ethayarajh, 2018)[5].

[5] Ethayarajh (2018) proposed three methods, i.e., S, U, and

| | ADD | W/O NORM |
|---|---|---|
| GloVe | **57.60** | 50.83 |
| word2vec | **72.22** | 62.97 |
| fastText | **71.49** | 59.25 |

(a) Pre-trained word vectors.

| | ADD | W/O NORM | diff |
|---|---|---|---|
| GloVe | **57.60** | 50.83 | 6.77 |
| + A | **67.56** | 59.03 | 8.53 |
| + AW | **76.06** | 59.03 | 17.03 |
| + **VC**(AWR) | **77.51** | 59.71 | 17.80 |

| | ADD | W/O NORM | diff |
|---|---|---|---|
| word2vec | **72.22** | 62.97 | 9.25 |
| + A | **72.32** | 63.37 | 8.95 |
| + AW | **76.61** | 63.37 | 13.24 |
| + **VC**(AWR) | **77.33** | 63.96 | 13.37 |

| | ADD | W/O NORM | diff |
|---|---|---|---|
| fastText | **71.49** | 59.25 | 12.24 |
| + A | **69.67** | 57.34 | 12.33 |
| + AW | **79.09** | 57.34 | 21.75 |
| + **VC**(AWR) | **79.68** | 57.24 | 22.44 |

(b) Converted word vectors.

Table 2: Norm has a large impact on additive composition. Pearson's $r \times 100$ between the predicted and gold scores is reported. The best result in each row is indicated in **bold**. The STS-B dataset (dev) was used.

- $\alpha_2$: 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 CCR (Ethayarajh, 2018), **C**onceptor removal (Liu et al., 2019b).

Henceforth, a bold character denotes each method. In addition, **VC**(AWR), for example, denotes **VC** induced by **A**, **W**, and **R**. Note that we did not tune hyperparameters; we used values reported in previous studies. See Appendix D for details.

### 7.1 Workings of Norm

Here, we experimentally confirm whether the norm of a word vector is in fact a good proxy of the word's in-sentence importance.

**Pre-trained Word Vectors.** Let us consider another additive composition than that in Equation 11, which excludes the effect of weighting by the norm

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

Table 2a shows the experimental results obtained using two types of sentence vectors (11), (25). Ignoring the norm of the word vectors produced consistently poor performance. This demonstrates that the norm of a word vector certainly plays the role of the weighting factor of the word.

**Converted Word Vectors.** To verify our hypothesis that VC improves the norm, we performed the same experiments as above using converted word vectors. Table 2b shows that as the word vectors are gradually converted, the difference in predictive performance between Equation 11 and 25 (i.e., the performance gain by norm) increased. This fact supports our hypothesis that VC "grows" the norm.

P. For the correctness, we abbreviate this series of methods as SUP, which was abbreviated as UP in the original paper.

[6] "+ AW" is omitted from Table 3b because W (i.e., the scaling function) alone does not change the angle.

| | L2 | DOT | COS |
|---|---|---|---|
| GloVe | | | |
| MEN | 73.36 | **80.79** | **80.49** |
| RW | 45.13 | 48.17 | **51.04** |
| word2vec | | | |
| MEN | 62.31 | 74.46 | **78.20** |
| RW | 35.10 | 51.29 | **55.80** |
| fastText | | | |
| MEN | 68.67 | 79.37 | **84.55** |
| RW | 47.24 | 53.83 | **62.17** |

(a) It is better to use only direction instead of the norm.

| | COS original vector | COS + A | COS + **VC** (AWR) |
|---|---|---|---|
| GloVe | | | |
| MEN | 80.49 | 82.58 | **83.69** |
| RW | 51.04 | **57.87** | **57.96** |
| word2vec | | | |
| MEN | 78.20 | **80.22** | **80.15** |
| RW | 55.80 | **57.39** | **57.47** |
| fastText | | | |
| MEN | 84.55 | **85.51** | **85.91** |
| RW | 62.17 | **62.98** | **62.75** |

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

Table 3: The angle of word vectors is a good proxy for word similarity. Spearman's $\rho \times 100$ between the predicted and gold scores is reported. In each row, the best result and results where the difference from the best result was $< 0.5$ are indicated in **bold**. [6]

### 7.2 Workings of Angle

We assumed the angle between two word vectors is a good proxy for the dissimilarity of two words. Presently, the cosine similarity between word vectors is a common 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, a question arises, i.e., which is the most suitable method to compute word dissimilarity? To answer this question, we compared dissimilarity metrics using nine word similarity datasets[7].

**Pre-trained Word Vectors.** Table 3a shows that cosine similarity (i.e., ignoring the norm) yields relatively higher correlation with human evaluations compared to dot product or Euclidean distance (i.e., using the norm). This indicates that the angle of

[7] See Appendix A for details regarding the datasets. While the results for the two larger datasets are presented here, the results for all nine datasets are presented in Appendix E.

| | GloVe | | word2vec | | fastText | |
|---|---|---|---|---|---|---|
| | WMD | **WRD** | WMD | **WRD** | WMD | **WRD** |
| original vector | 65.74 | **67.70** | 67.21 | **70.91** | 64.06 | **69.31** |
| + A | 65.44 | **68.26** | 67.09 | **71.23** | 63.79 | **69.34** |
| + AW | 64.76 | **76.08** | 64.97 | **75.00** | 62.47 | **76.90** |
| + **VC**(AWR) | 64.74 | **76.87** | 64.89 | **76.04** | 62.47 | **77.56** |
| + SIF weights | 75.42 | - | 73.90 | - | 74.64 | - |

Table 4: The combination of WRD and VC gave the best performance. Pearson's $r \times 100$ between the predicted and gold scores is reported. The STS-B dataset (dev) was used. The best result and results where the difference from the best $< 0.5$ in each row are in **bold**, and the best results are further **underlined**.

word vectors encodes the dissimilarity of words relatively well; in contrast, the norm is not relevant.

**Converted Word Vectors.** In view of the discussion given in Section 6, we expected that the word dissimilarity of $w$ and $w'$ would be better encoded in the angle between the converted word vectors $\langle \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{u}}' \rangle = \cos(\widetilde{\boldsymbol{w}}, \widetilde{\boldsymbol{w}}')$ than that between the original word vectors $\langle \boldsymbol{u}, \boldsymbol{u}' \rangle = \cos(\boldsymbol{w}, \boldsymbol{w}')$. Table 3b shows that, as the word vectors were gradually converted, the angle of word vectors became more accurate as a measure of the dissimilarity of words.

### 7.3 Ablation Study

We experimentally confirmed the effectiveness of each WRD and VC, through the degree of performance improvement over the baseline WMD. Table 4 shows the results. In nearly all cases, WRD demonstrated higher performance than WMD. We summarize some major findings as follows.

- The performance of WRD improves steadily, as word vectors are transformed by VC, 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.
- One may consider that W (SIF weighting) can be used directly as the probability mass for WMD because it is simply a scaling factor for each word. "+ SIF weights" in Table 4 represents such a computation; however, even when WMD and employed SIF directly, it did not reach the performance of WRD and VC.

Following Kusner et al. (2015), we further experimented with stopword removal. Stopword removal was a good heuristic that gave both WMD and WRD a large performance gain similar to SIF; however, the above two findings remained unchanged. See Appendix E for additional details.

| | 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) | ***77.63*** | ***73.14*** | ***56.81*** |
| 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) | ***77.85*** | *74.94* | ***58.22*** |
| Sent2Vec (Pagliardini et al., 2018) | - | 75.5* | - |
| Skip-Thought‡ (Kiros et al., 2015) | 46 | - | - |
| ELMo ‡ (Peters et al., 2018) | 68 | - | - |

Table 5: Pearson's $r \times 100$ between the predicted and gold scores is shown. The best results in each dataset, word vector, and strategy for computing the textual similarity ("Additive composition" or "Considering Word Alignment") are in **bold**; and the best results regardless of the strategy are further **underlined**. Each row marked (†) was re-implemented by us. Each value marked (‡) was taken from Perone et al. (2018), and marked (∗) was taken from STS Wiki[8].

### 7.4 Benchmark Tasks

Finally, we compared the performance of the proposed WRD and VC methods to that of various baselines, including recent alignment-based methods, i.e., **WMD** (Kusner et al., 2015), **BERTScore** (Zhang et al., 2019)[9], and **DynaMax** (Zhelezniak et al., 2019). The results are shown in Table 5. We summarize our major findings as follows.

- Among the methods that consider word alignment, WRD + VC achieved the best performance. This is likely due to the fact that other meth-

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

[9] BERTScore was used as an STS method. We did not use BERT itself. See Appendix B for details.

ods employ Euclidean distance (WMD) or dot product (DynaMax) as word similarity measures. These metrics cannot distinguish the two types of information (i.e., weight and dissimilarity). BERTScore applies cosine similarity like WRD; however, BERTScore was inferior to WRD on average, which can be attributed to the fact that BERTScore completely disregards the norm.

- Compared to strong baselines based on additive composition (+WR, +SUP), WRD using the same word vectors (+**VC**(WR), +**VC**(WR)) performed equally or better. This result was unexpected given that +WR and +SUP were originally proposed to create sentence vectors, and WRD simply uses them without tuning. Thus, we believe that considering word alignment is an inherently good principle for STS.

Refer to Appendix E for the more comprehensive results obtained using additional datasets and methods, including (semi-)supervised approaches.

## 8 Connection to Other Methods

Finally, we discuss the relationships among *WRD*, *WMD*, and cosine similarity of additive composition (11, 12), which we refer to as *ADD*, from a sentence representation perspective.

**Connection to Additive Composition.** Surprisingly, ADD is a special case (i.e., a simplified version) of WRD. In fact, given a discrete-distribution representation containing only a single sentence vector, i.e., $\boldsymbol{\mu}_s^{\text{point}} = \{(\boldsymbol{s}, 1)\}$, the obvious EMD cost using cosine distance is equivalent to ADD.

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

This relationship between ADD and WRD becomes clearer when examining their sentence representations using the norm ($\lambda_i$) and direction vector ($\boldsymbol{u}_i$):

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

where $Z := \sum_i \lambda_i$, and $\delta[\cdot]$ is the Dirac delta function. Initially, they appear quite similar. However, the key difference is 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. Given that STS tasks require word alignment (where words are treated disjointly), it is natural that WRD (where word vectors are treated disjointly) achieves better performance on STS tasks[10].

[10] In contrast, we have confirmed that ADD demonstrated higher performance than WRD on the topic similarity task (SICK-R. See Appendix E for details). For a task where it is sufficient to know the trend of the meaning of the entire sentence, it may 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 represent sentences as bag-of-word-vectors representations? Sentence representations for ADD and WMD are as follows:

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

The barycenters (27a), (28a) for ADD are identical up to scale because $\lambda_i \boldsymbol{u}_i = \boldsymbol{w}_i$ holds. In contrast, the discrete distributions (27b) for WRD and (28b) for WMD are quite different. WRD treats the norm $\lambda$ as a weighting factor, as ADD implicitly does (28a). In contrast, WMD assigns uniform weights to both long and short vectors (28b), which is one reason the most natural representation (28b) employed by WMD does not work effectively.

The difference in performance between WMD and WRD can also be explained by the difference in the transportation cost functions. Many word embeddings use inner product as the training objective function, i.e., the origin of the embedding space is meaningful. Also, cosine distance used in WRD depends on the position of the origin. In contrast, parallel translation invariant Euclidean cost used in WMD ignores the position of the origin.

## 9 Conclusion

In this paper, we first indicated (i) that the *norm* and *angle* of word vectors are good proxies for the importance of a word and dissimilarity between words, respectively, and (ii) that some previous alignment-based STS methods inappropriately "mix up" them. With these findings, we have proposed word rotator's distance (WRD), which is a new unsupervised, EMD-based STS metric. WRD was designed so that the norm and angle of word vectors correspond to the probability mass and transportation cost in EMD, respectively. In addition, we found that the latest powerful sentence-vector estimation methods implicitly improve the norm and angle of word vectors, and we can exploit this effect as a word vector converter (VC). In experiments on multiple STS tasks, the proposed methods outperformed not only alignment-based methods such as word mover's distance, but also powerful addition-based sentence vectors.

## 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 JST CREST, Grant Number JPMJCR1513, Japan. This work was also supported by JSPS KAKENHI, Grant Numbers 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.

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.

Rémi Flamary and Nicolas Courty. 2017. POT Python Optimal Transport library.

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.

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.

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 English pre-trained word embeddings in our experiments.

- **GloVe** trained with Common Crawl (Pennington et al., 2014)[11]
- **word2vec** trained with Google News (Mikolov et al., 2013)[12]
- **fastText** trained with Common Crawl (Bojanowski et al., 2017)[13]
- **PSL**, the ParagramSL-999 embeddings, trained with the PPDB paraphrase database (Wieting et al., 2015)[14]
- **ParaNMT** trained with ParaNMT-50, a large scale paraphrase database (Wieting and Gimpel, 2018)[15]

## A.2 Word Similarity Datasets

We used the following nine word similarity tasks in our experiments.

- **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

We used the following English 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)[16]

[11] https://nlp.stanford.edu/projects/glove/
[12] https://code.google.com/archive/p/word2vec/
[13] https://fasttext.cc/docs/en/english-vectors.html
[14] http://www.cs.cmu.edu/~jwieting/
[15] https://github.com/kawine/usif
[16] https://github.com/cocoxu/SemEval-PIT2015

- **SICK-R**: SemEval 2014 semantic relatedness task (Marelli et al., 2014)

**Tokenization.** In each experiment, we first tokenized all the STS datasets (besides the Twitter dataset) by NLTK (Bird and Loper, 2004) with some post-processing steps following Ethayarajh (2018)[17]. The Twitter dataset has already been tokenized by the workshop organizer. We then lowercased all tokens to conduct experiments under the same conditions with cased embeddings and non-cased embeddings.

### A.4 Stopword List

The stopword list based on the SMART Information Retrieval System[18] was used for WMD (Kusner et al., 2015) and conceptor removal (C) (Liu et al., 2019a).

## 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[19] show that BERTScore with pre-trained BERT/RoBERTa performs poorly on unsupervised STS.

Nonetheless, BERTScore is definitely promising as a method. We then reported the results of BERTScore using *non*-contextualized word vectors, e.g., GloVe, and we confirmed higher performance compared to using pre-trained BERT. Needless to say, the application of BERT-based models to unsupervised STS is an important future research topic.

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

[18] https://github.com/igorbrigadir/stopwords

[19] We used BERT-large and RoBERTa-large. For embeddings, 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 $s = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$, $s' = (\boldsymbol{w}'_1, \ldots, \boldsymbol{w}'_{n'})$

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: $\boldsymbol{m}_s \leftarrow \frac{1}{Z}(\|\boldsymbol{w}_1\|, \ldots, \|\boldsymbol{w}_n\|) \in \mathbb{R}^n$
4: $\boldsymbol{m}_{s'} \leftarrow \frac{1}{Z'}(\|\boldsymbol{w}'_1\|, \ldots, \|\boldsymbol{w}'_{n'}\|) \in \mathbb{R}^{n'}$
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[20]. 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 $f_{\mathrm{VC}}$ (Equation 22).

When computing Algorithm 2, we set hyperparameters as

- $D_{\mathrm{A}} = 3$ for **A** (Mu and Viswanath, 2018)
- $a_{\mathrm{W}} = 10^{-3}$ for **W** (Arora et al., 2017)
- $a_{\mathrm{U}} \approx 1.2 \times 10^{-3}$ for **U** (Ethayarajh, 2018)
- $D_3 = 1$ for **R** (Arora et al., 2017)
- $D_3 = 5$ for **P** (Ethayarajh, 2018)
- $D_{\mathrm{C}} = 1$ for **C** (Liu et al., 2019a)

reported in previous studies, without any additional tuning.

Prior to performing all-but-the-top (A), we restricted the vocabulary of word vectors to words

[20] In our experiments, we used the well-developed python optimal transport (POT) library (Flamary and Courty, 2017): https://github.com/rflamary/POT/. In particular, `ot.emd2()` was used.

appearing more than 200 times in the enwiki corpus[21], following Liu et al. (2019b).

We used the unigram probability $\mathbb{P}$ of English words estimated using the enwiki dataset, preprocessed by Arora et al. (2017)[22].

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.

## E Full Results of Experiments

### E.1 Workings of Angle

See Table 7 for full results using nine word similarity datasets.

### E.2 Ablation Study

See Table 8 for full results. WRD *without* stopword removal achieves the best results. This is likely because WRD can more continuatively compare the differences in the importance between stopwords using their norm.

### E.3 Benchmark Tasks

See Table 9 for full results in an unsupervised settings. See Table 10 for full results in an semisupervised and supervised settings.

[21] https://github.com/PrincetonML/SIF/. Preprocessed by Arora et al. (2017).

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

**Algorithm 2** Vector Converter (VC), induced from **A**ll-but-the-top (Mu and Viswanath, 2018), SIF **W**eighting (Arora et al., 2017) or **U**nsupervised SIF (Ethayarajh, 2018), and common component **R**emoval (Arora et al., 2017) or **P**iecewise CCR (Ethayarajh, 2018) or **C**onceptor removal (Liu et al., 2019a) .

**Input:** pre-trained word vectors $\{\boldsymbol{w}_1, \dots, \boldsymbol{w}_{|\mathcal{V}|}\}$, sentences in interest $\mathcal{S} = \{s_1, \dots, s_{|\mathcal{S}|}\}$, word probability $\mathbb{P}\colon\ \mathcal{V} \to [0,1]$, stopwords $\mathcal{V}_{\mathrm{SW}} \subseteq \mathcal{V}$, and constants $D_{\mathrm{A}}$, $a_{\mathrm{W}}$ (or $a_{\mathrm{U}}$), $D_3$ (or $D_{\mathrm{C}}$)

Compute parameters of $f_1$:

$\cdots$ if using **A**ll-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

$\boldsymbol{A}_1 \leftarrow \boldsymbol{I} - \sum_{j=1}^{D_{\mathrm{A}}} \boldsymbol{u}_j \boldsymbol{u}_j^\top \in \mathbb{R}^{d \times d}$

$\boldsymbol{b}_1 \leftarrow \overline{\boldsymbol{w}} \in \mathbb{R}^d$

Compute parameters of $\alpha_2$:

**for each** $w$ **do**

  $\cdots$ if using SIF **W**eighting:

  $\alpha_2(w) \leftarrow a_{\mathrm{W}}/(\mathbb{P}(w) + a_{\mathrm{W}}) \in \mathbb{R}$

  $\cdots$ else if using **U**nsupervised SIF:

  $\alpha_2(w) \leftarrow a_{\mathrm{U}}/(\mathbb{P}(w) + \frac{1}{2}a_{\mathrm{U}}) \in \mathbb{R}$

**end for**

Create sentence vectors temporarily:

**for** $i \leftarrow 1$ to $|\mathcal{S}|$ **do**

  $\boldsymbol{s}_i \leftarrow \frac{1}{|s_i|}\sum_{w \in s_i} \alpha_2(w) \boldsymbol{A}_1(\boldsymbol{w} - \boldsymbol{b}_1) \in \mathbb{R}^d$

**end for**

Compute parameters of $f_3$:

$\cdots$ if using CC**R** or **P**iecewise CCR:

$(\boldsymbol{v}_1, \sigma_1), \dots, (\boldsymbol{v}_{D_3}, \sigma_{D_3}) \leftarrow \mathrm{PCA}(\{\boldsymbol{s}, \dots, \boldsymbol{s}_{|\mathcal{S}|}\})$ ▷ top $D_3$ singular vectors and singular values

$$\boldsymbol{A}_3 \leftarrow \boldsymbol{I} - \sum_{i=1}^{D_3} \frac{\sigma_i^2}{\sum_{j=1}^{D_3}\sigma_j^2} \boldsymbol{v}_i \boldsymbol{v}_i^\top \in \mathbb{R}^{d \times d}$$

$\cdots$ else if using **C**onceptor removal:

$\boldsymbol{R} \leftarrow \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \boldsymbol{s}_i \boldsymbol{s}_i^\top \in \mathbb{R}^{d \times d}$

$\boldsymbol{C} \leftarrow \boldsymbol{R}(\boldsymbol{R} + \alpha_{\mathrm{C}}^{-2}\boldsymbol{I})^{-1} \in \mathbb{R}^{d \times d}$

$\boldsymbol{R}_{\mathrm{SW}} \leftarrow \frac{1}{|\mathcal{V}_{\mathrm{SW}}|}\sum_{w \in \mathcal{V}_{\mathrm{SW}}} \boldsymbol{w}\boldsymbol{w}^\top$

$\boldsymbol{C}_{\mathrm{SW}} \leftarrow \boldsymbol{R}_{\mathrm{SW}}(\boldsymbol{R}_{\mathrm{SW}} + \alpha_{\mathrm{C}}^{-2}\boldsymbol{I})^{-1}$

$\boldsymbol{A}_3 \leftarrow ((\boldsymbol{I} - \boldsymbol{C})^{-1} + (\boldsymbol{I} - \boldsymbol{C}_{\mathrm{SW}})^{-1} - \boldsymbol{I})^{-1}$

Convert word vectors:

**for** $i \leftarrow 1$ to $|\mathcal{V}|$ **do**

  $\widetilde{\boldsymbol{w}}_i \leftarrow \boldsymbol{A}_3(\alpha_2(w)\boldsymbol{A}_1(\boldsymbol{w}_i - \boldsymbol{b}_1))$

**end for**

**Output:** Converted word vectors $\{\widetilde{\boldsymbol{w}}_1, \cdots, \widetilde{\boldsymbol{w}}_{|\mathcal{V}|}\}$

| | $f_1$ denoising word vectors | $\alpha_2$ scaling | $f_3$ 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) | Sentence-wise Feature Scaling | uSIF (Unsupervised SIF) | Piecewise CCR |
| Liu et al. (2019b) | Conceptor Negation | – | – |
| Liu et al. (2019a) | – | SIF | Conceptor Removal |

Table 6: Sentence-vector estimation methods.

| | L2 | DOT | COS |
|---|---|---|---|
| GloVe | | | |
| MEN | 73.36 | **80.79** | **80.49** |
| MTurk287 | 60.87 | **69.50** | **69.18** |
| MC30 | 75.22 | 76.77 | **78.81** |
| RW | 45.13 | 48.17 | **51.04** |
| RG65 | 70.75 | **77.79** | 76.90 |
| SCWS | 56.25 | 62.20 | **63.29** |
| SimLex999 | 35.03 | 38.86 | **40.71** |
| WS353-REL | 49.74 | **72.35** | 68.75 |
| WS353-SIM | 69.03 | **79.54** | **79.57** |
| word2vec | | | |
| MEN | 62.31 | 74.46 | **78.20** |
| MTurk287 | 49.43 | 66.66 | **68.37** |
| MC30 | 69.88 | 76.57 | **78.87** |
| RW | 35.10 | 51.29 | **55.80** |
| RG65 | 71.30 | 72.58 | **76.17** |
| SCWS | 47.46 | 65.10 | **66.49** |
| SimLex999 | 32.35 | 43.23 | **44.20** |
| WS353-REL | 40.74 | 56.65 | **61.40** |
| WS353-SIM | 55.82 | 74.89 | **77.39** |
| fastText | | | |
| MEN | 68.67 | 79.37 | **84.55** |
| MTurk287 | 56.74 | 66.10 | **72.47** |
| MC30 | 77.11 | 81.36 | **85.24** |
| RW | 47.24 | 53.83 | **62.17** |
| RG65 | 79.72 | 80.58 | **86.26** |
| SCWS | 52.88 | 64.90 | **69.42** |
| SimLex999 | 36.91 | 46.71 | **50.47** |
| WS353-REL | 49.44 | 63.64 | **72.33** |
| WS353-SIM | 66.83 | 79.34 | **84.24** |

(a) It is better to use only direction instead of the norm.

| | COS original vector | COS + A | COS + **VC** (AWR) |
|---|---|---|---|
| GloVe | | | |
| MEN | 80.49 | 82.58 | **83.69** |
| MTurk287 | 69.18 | **73.11** | 72.73 |
| MC30 | **78.81** | 77.29 | **78.51** |
| RW | 51.04 | **57.87** | **57.96** |
| RG65 | **76.90** | 76.30 | 76.28 |
| SCWS | 63.29 | **66.65** | **66.48** |
| SimLex999 | 40.71 | **46.55** | **47.02** |
| WS353-REL | 68.75 | **72.27** | **72.14** |
| WS353-SIM | 79.57 | **81.19** | 80.29 |
| word2vec | | | |
| MEN | 78.20 | **80.22** | **80.15** |
| MTurk287 | 68.37 | **65.21** | **65.20** |
| MC30 | 78.87 | 81.06 | **82.20** |
| RW | 55.80 | **57.39** | **57.47** |
| RG65 | 76.17 | **82.22** | **82.57** |
| SCWS | 66.49 | **65.08** | **65.23** |
| SimLex999 | 44.20 | **46.23** | **46.53** |
| WS353-REL | 61.40 | **60.96** | **61.07** |
| WS353-SIM | 77.39 | **77.07** | **77.18** |
| fastText | | | |
| MEN | 84.55 | **85.51** | **85.91** |
| MTurk287 | **72.47** | **71.25** | 71.02 |
| MC30 | 85.24 | **86.74** | **86.69** |
| RW | 62.17 | **62.98** | **62.75** |
| RG65 | 86.26 | 88.58 | **89.33** |
| SCWS | **69.42** | **69.07** | 68.62 |
| SimLex999 | 50.47 | **51.83** | **52.26** |
| WS353-REL | **72.33** | **72.32** | **72.81** |
| WS353-SIM | **84.24** | **84.69** | **84.46** |

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

Table 7: The angle of word vectors is a good proxy for word similarity (full results). Spearman's $\rho \times 100$ between the predicted and gold scores is reported. In each row, the best result and results where the difference from the best result was $< 0.5$ are indicated in **bold**. "+ AW" is omitted from Table 3b because W (i.e., the scaling function) alone does not change the angle.

| Removing Stopwords | WMD | **WRD** | WMD ✓ | **WRD** ✓ |
|---|---|---|---|---|
| GloVe | 65.74 | **67.70** | **75.42** | 74.43 |
| GloVe + A | 65.44 | **68.26** | **74.78** | **74.68** |
| GloVe + AW | 64.76 | **76.08** | 74.53 | **75.62** |
| GloVe + **VC**(AWR) | 64.74 | **76.87** | 74.47 | **76.49** |
| GloVe + SIF weights | 75.42 | - | 76.41 | - |
| GloVe + A + SIF weights | 75.59 | - | 75.75 | - |
| word2vec | 67.21 | **70.91** | 72.40 | **73.11** |
| word2vec + A | 67.09 | **71.23** | 72.31 | **73.46** |
| word2vec + AW | 64.97 | **75.00** | 71.88 | **74.25** |
| word2vec + **VC**(AWR) | 64.89 | **76.04** | 71.71 | **75.39** |
| word2vec + SIF weights | 73.90 | - | 73.41 | - |
| word2vec + A + SIF weights | 73.76 | - | 73.27 | - |
| fastText | 64.06 | **69.31** | 73.82 | **75.65** |
| fastText + A | 63.79 | **69.34** | 73.32 | **75.72** |
| fastText + AW | 62.47 | **76.90** | 72.89 | **76.54** |
| fastText + **VC**(AWR) | 62.47 | **77.56** | 72.85 | **77.26** |
| fastText + SIF weights | 74.64 | - | 74.79 | - |
| fastText + A + SIF weights | 74.28 | - | 74.28 | - |

Table 8: The combination of WRD and VC gave the best performance (full results). Pearson's $r \times 100$ between the predicted and gold scores is reported. The STS-B dataset (dev) was 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**.

| | 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) | *64.22* | *58.07* | *71.58* | *77.63* | *73.14* | *56.81* | ***67.79*** |
| **WRD** GloVe + **VC**(SUC) | ***64.39*** | ***58.41*** | ***72.00*** | ***77.76*** | ***74.16*** | ***57.11*** | *67.07* |
| 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 | 39.53 | 64.05 |
| 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) | ***61.95*** | *55.47* | *69.18* | ***75.69*** | *71.59* | *43.99* | *66.53* |
| **WRD** word2vec + **VC**(SUC) | *61.95* | *55.64* | ***69.36*** | *75.56* | *72.29* | ***44.17*** | *65.54* |
| 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) | *64.07* | *58.75* | *71.59* | ***77.85*** | *74.94* | ***58.22*** | *67.83* |
| **WRD** fastText + **VC**(SUC) | *64.00* | *59.06* | ***71.81*** | *77.77* | *75.56* | *57.98* | *67.01* |
| 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 9: Pearson's $r \times 100$ between the predicted scores and the gold scores is shown. The best results in each block 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 (†) was re-implemented by us. Each value marked (‡) was taken from Perone et al. (2018). Each value marked (∗) was taken from STS Wiki (`http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark`).

| | 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 10: Pearson's $r \times 100$ between the predicted and gold scores is show. 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 (†) was re-implemented by us. Each value marked (‡) was taken from Perone et al. (2018). Each value marked (∗) was taken from STS Wiki (http://ixa2.si.ehu.es/stswiki/index.php/STSbenchmark).