How Do Simple Transformations of Text and Image Features Impact Cosine-Based Semantic Match?

Guillem Collell(✉) and Marie-Francine Moens

Department of Computer Science, KU Leuven, 3001 Heverlee, Belgium
{gcollell,sien.moens}@kuleuven.be2

Abstract. Practitioners often resort to off-the-shelf feature extractors such as language models (e.g., BERT or Glove) for text or pre-trained CNNs for images. These features are often used without further supervision in tasks such as text or image retrieval and semantic similarity with cosine-based semantic match. Although cosine similarity is sensitive to centering and other feature transforms, their impact on task performance has not been systematically studied. Prior studies are limited to a single domain (e.g., bilingual embeddings) and one data modality (text). Here, we systematically study the effect of simple feature transforms (e.g., standardizing) in 25 datasets with 6 tasks covering semantic similarity and text and image retrieval. We further back up our claims in *ad-hoc* laboratory experiments. We include 15 (8 image + 7 text) embeddings, covering the state-of-the-art models. Our second goal is to determine whether the common practice of defaulting to the cosine similarity is empirically supported. Our findings reveal that: (i) some feature transforms provide solid improvements, suggesting their default adoption; (ii) cosine similarity fares better than Euclidean similarity, thus backing up standard practices. Ultimately, our takeaways provide actionable advice for practitioners.

Keywords: Feature transform · Cosine similarity · Image retrieval · Text retrieval · Semantic similarity · Text embeddings · Image embeddings

1 Introduction

Extraction of image and text features with pre-trained off-the-shelf models enjoys widespread adoption among practitioners (e.g., BERT-as-a-service [58]). These features are often used in tasks such as multimedia retrieval [50,55], semantic similarity [12,13,26,40,52], word analogies [37,40] or zero-shot image recognition [46,61], to name a few. Not infrequently, features are used directly without further supervised training, typically via cosine-based semantic match. As noted [1,24], the cosine similarity is sensitive to centering, cross-dimension correlations and scale variations (Fig. 1). However, the extent to which this impacts task performance has not yet been systematically studied. Studies assessing the effect

D. Hiemstra et al. (Eds.): ECIR 2021, LNCS 12656, pp. 98–114, 2021.
https://doi.org/10.1007/978-3-030-72113-8_7

of feature transforms (e.g., normalizing or PCA) typically restrict to a single domain and task (e.g., bilingual word embeddings [1,59]) and a single modality (text). This prompts our first research question (**RQ1**): *Can we improve the features with simple transforms in a variety of text and image tasks?* In particular, quantifying the (hypothesized) negative impact of vector *uncenteredness* on cosine-based performance (Fig. 1) is among our foremost hypothesis to test.

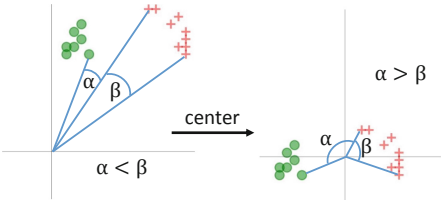

Fig. 1. Illustration of uncentered vectors hindering cosine similarity performance. Since cosine similarity computes the angle (α, β) from the origin $\vec{0}$, in this example where all vectors are dimension-wise positive, the *cosine* judges two points from different classes as more similar than two points of the same class. *Centering* helps obtaining more meaningful similarity estimates.

The cosine is generally chosen as default similarity measure in retrieval [15,50] and semantic similarity tasks [12,26,27,31,40,52,53]. This choice may eventually be informed in a (labelled) validation set or even the metric itself can be learned [14,56] if a labelled training set exists. However, because often none of these are available [12,26,31,40,46,52], our study assumes a scenario without either set. This motivates our second research question (**RQ2**): *Is the default choice of cosine similarity (versus Euclidean) empirically supported?*

To answer **RQ1** and **RQ2**, we perform an extensive empirical study in *real-world* tasks with both image and text data. We provide further insight and back up our claims in laboratory experiments. Our tests include 25 datasets with 6 different tasks covering text and image retrieval, word-, sentence- and visual-similarity, and paraphrase detection. We include 15 types of image (8) and text (7) embeddings, covering state-of-the-art models. Simple feature transforms are also compared with manifold learning methods.

Our findings reveal that: (i) *Centering* and *standardizing* are remarkably effective across *real-world* tasks (**RQ1**); (ii) the *cosine* significantly outperforms the *Euclidean* similarity across 74 conditions (embedding × task), hence supporting the default choice (**RQ2**). Ultimately, our findings provide actionable advice to practitioners and warning about the negative impact of using cosine similarity along with uncentered features.

This paper is organized as follows. In Sect. 2, we discuss related work. We present our methods in Sect. 3 and our tasks in Sect. 4. In Sect. 5, we describe our embeddings and setup. In Sect. 6, we discuss our empirical results. Section 7 concludes the paper.

2 Related Work

Feature transforms: [25] study the optimality of five different whitening transformations from the viewpoint of the properties of their covariance matrices.

In contrast to this study, [25] do not include empirical evaluations in text or image problems.

Additionally, [11] studied the effect of transforming features with an untrained neural network (i.e., random projections), finding that the performance of transformed vectors does not drop in word-similarity tasks. The impact on performance of different feature transforms on *classification* problems such as of biomedical data is also studied [4].

The closest works to ours are [32], [59] and [1], all of whom study the effect of feature transforms in the context of text problems. [32] study the effect of hyperparameters and normalization of word embeddings, revealing that the impact of design decisions and hyperparameters on performance is more important than the choice of the embedding algorithms themselves. [59] finds that constraining word embeddings to the unit hyper-sphere (i.e., normalizing them) improves performance in mono-lingual word similarity and bi-lingual word translation. [1] investigate several transformations including PCA, mean centering, normalization and whitening in the context of multi-lingual word embeddings. In contrast with ours, these studies restrict to a single domain and to text data (no images), and do not discuss *standardizing* – which we find to be a top performer.

Similarity measures: [24] analytically study the behavior and properties of similarity measures such as cosine similarity and the inner product from a geometric viewpoint, focusing on iso-similarity contours. Also analytically, [41] studies similarity measures in the retrieval context. In contrast to them, we carry out extensive empirical tests.

Metric learning: algorithms such as the ITML [14] or LMNN [56] learn a metric distance which can be seen as a form of learning a suitable transformation to the input vectors. However, this metric is learned in a supervised fashion, typically to be used in conjunction with a nearest-neighbor classifier, which falls out of the *unsupervised* scope of our study. It is worth mentioning that unsupervised metric learning algorithms also exist [9,23], yet they do not witness widespread adoption among data practitioners.

Manifold learning: methods, such as Isomap [49], Locally Linear Embedding (LLE) [42], diffusion maps [10], multi dimensional scaling (MDS) [29] or t-SNE [34], try to discover the underlying data manifold, which enables disentangling the vectors in a lower-dimensional space. Such methods are widely used for data visualization, yet they are not popular as feature transforms for predictive models – perhaps due to their limited success for such purpose. Although the inclusion of manifold learning methods in our study obeys mainly completeness reasons – given that our focus are *simple* feature transforms – an empirical comparison of *simple* transforms and manifold learning methods across multiple tasks has not been performed yet and we believe that is of practical interest.

3 Method

Let us first lay down our general **framework**. Let $S = \{s_i\}_{i=1}^{N}$ be a set of N data points (sentences, words or images). One extracts corresponding feature vectors

$V = \{v_i\}_{i=1}^N$ with a text or image encoder $E()$ (e.g., BERT or a CNN model), where $v_i = E(s_i)$ and $v_i \in \mathbb{R}^d$. The parameters θ of a feature transform T_θ are learned using the vectors V (e.g., in *centering*, θ are the dimension-wise means). A new vector v can then be transformed with $T_\theta(v)$ (Sect. 3.1), where v may belong or not to the set V used for learning T_θ.

3.1 Feature Transforms

In the following, we describe the feature transforms included in our experiments.

- **Original (orig):** denotes the original vectors $V = \{v_i\}_{i=1}^N$ without any transformation.
- **Centering (ctr):** $\text{ctr}(v) = v - \overline{V}$; subtracts the centroid vector $\overline{V} = \frac{1}{N}\sum_{i=1}^N v_i = \frac{1}{N}\sum_{i=1}^N (v_i^1, \cdots, v_i^d)$ to a vector v.
- **Standardizing (stz):** $\text{stz}(v) = (v - \overline{V})/\text{sd(V)}$; where sd(V) are the component-wise standard deviations $\text{sd(V)} = (\text{sd}(V^1), \cdots, \text{sd}(V^d))$ with $V^k = \{v_i^k\}_{i=1}^N$; and sd() is the standard deviation. *Stz* zero-means the data V and sets variances equal to 1.
- **Whitening (wht):** We use the *Zero Components Analysis* (ZCA) whitening as described in [28]. ZCA de-correlates the data dimensions and makes the variances equal to 1.
- **Normalizing (Nrm):** $\text{nrm}(v) = v/||v||$; moves any vector v to the unit hypersphere. Unlike the rest, this transform depends only on the same vector v, and not on the whole set $V = \{v_i\}_{i=1}^N$. Normalizing has no effect when the *cosine* similarity is used.
- **Isomap (Iso):** [49] and **Locally Linear Embedding (LLE)** [42] are used analogously by first learning the parameters θ in the training set of vectors V, and applying the learned transformation T_θ to a new vector $v \in \mathbb{R}^d$, with $T_\theta(v) \in \mathbb{R}^m$ with $m \leq d$[1].
- **Principal Component Analysis (PCA):** is a classical dimensionality reduction method that finds orthogonal directions that best fit the data in the least-squares sense. We keep a number of components (dimensions) such that 80% of the variance is explained.[2] Our implementation of PCA [39] centers but does not scale the data (for each feature) before applying the SVD decomposition.

Unlike simple transforms (e.g., *center*) the more complex PCA, Isomap and LLE have hyperparameters (e.g., output dimensionality) that impact their performance. Thus a validation set is often necessary, which is a shortcoming in our *unsupervised* setting.

[1] For both *Isomap* and *LLE* we set $m = 100$ in the *real-world*, and $m = 2$ for *synthetic* tasks. The number of nearest neighbors is set to 10 in all tasks (as default in sklearn [39]).

[2] The choice of 80% of the variance is discussed and compared to other values in the Supplement.

3.2 Complementary Experiments

- **Additive bias:** As a complement to *centering*, we study the effect of "uncenteredness" on the *cosine* similarity (as the Euclidean is shift invariant) by uncentering $V = \{v_i\}_{i=1}^N$ with a dimension-wise bias $b > 0$, namely $(v_i^1 + b, \cdots, v_i^d + b)\ \forall\ i = 1, \cdots, N$. This equates shifting all vectors to the positive quadrant (i.e., $v_i^k > 0\ \forall\ k = 1, \cdots, d$), if b large enough, or moving them further up in case they already are (see Sect. 6 for a discussion).
- **Multiplicative bias:** to study the effect of *non-homogeneity* of *scale* and *variances* across dimensions, we multiply each dimension with a bias $b > 0$ randomly drawn from a uniform $b \sim \mathcal{U}[0.001, 10]^d$, i.e., $v_i = (b_1 v_i^1, \cdots, b_d v_i^d)\ \forall\ i = 1, \cdots, N$. This study complements the *standardizing* method.

4 Tasks and Data

In this section, we first describe the procedure of two grand groups of tasks (Sect. 4.1), and then we introduce the datasets used in each individual task (Sect. 4.2). Our dataset selection criteria included: (i) Feasibility of implementing an *unsupervised* prediction approach (i.e., simply cosine-based); (ii) medium-sized datasets; (iii) rather popular and already clean data (thus little preprocessing required); (iv) diversity.

4.1 Task Descriptions

Grouping tasks: It is convenient to group our tasks in two functionally different categories, as they exhibit identical prediction-evaluation pipelines: (1) **Retrieval tasks:** (i) *text retrieval* and (ii) *image retrieval*; (2) **Similarity tasks:** (iii) *word similarity*, (iv) *sentence similarity*, (v) *visual similarity* and (vi) *synthetic data*. Furthermore, we refer throughout to **real-world** tasks being all tasks except the **synthetic** ones.

In *all* tasks, we consider two **similarity measures** to compute the *predicted similarity* $\text{sim}(s_1, s_2)$ between any two inputs s_1, s_2 (words, sentences or images) encoded with their respective features $v_i, v_j \in \mathbb{R}^d$:

- **Cosine** similarity: $\cos(v_i, v_j) = \frac{v_i v_j}{\|v_i\| \|v_j\|}$.
- **Euclidean** similarity $\text{Eucl}(v_i, v_j) = \frac{1}{1+\|v_i - v_j\|}$.

In the interest of the practitioner, we focus on *simple* and widely adopted transforms, *cosine* and Euclidean similarity, rather than aiming for an exhaustive comparison of all existing similarity measures and feature transforms. After having obtained the vectors $V = \{v_i\}_{i=1}^N$ and learned $T_\theta(v)$ as described in Sect. 3, we consider the task-specific procedures below.

■ Similarity tasks: All word-, sentence- and image-similarity datasets consist of a list of word, sentence or image pairs (s_i, s_j), e.g., ('car', 'truck') along with a

human (ground-truth) rating of their similarity or relatedness $y_{i,j} \in [1, 10]$. The system needs to predict a similarity score $\hat{y}_{i,j} \in [1, 10]$ for each pair (s_i, s_j). Model predictions are computed via $\cos(v_i, v_j)$ or $\text{Eucl}(v_i, v_j)$, where $(v_i, v_j) = E(s_i, s_j)$.

– **Evaluation:** Following [12,26,40], we use the **Spearman correlation** $\rho(\hat{y}, y)$ between the predicted $\hat{y} \in \mathbb{R}_+^N$ and the ground-truth similarity scores $y \in \mathbb{R}_+^N$ as the standard measure to evaluate the quality of semantic similarity predictions.

■ Retrieval tasks: We split the given test set V^{ts} into two disjoint sets: a *query set* $\mathcal{Q}$ and a *test collection* $\mathcal{T}$. Given a query $s_i \in \mathcal{Q}$, the goal of the task is to rank the relevant items from $\mathcal{T}$ higher than the non-relevant ones. The similarity between each item $s_i \in \mathcal{Q}$ in the query set $\mathcal{Q}$ is computed against *every* item $s_j \in \mathcal{T}$ in the test collection $\mathcal{T}$ via $\cos(v_i, v_j)$ or $\text{Eucl}(v_i, v_j)$ similarity, where $(v_i, v_j) = E(s_i, s_j)$.

– **Evaluation:** Performance is evaluated with the TREC standard **mean average precision (mAP)**, as described in [35]. Following [50,54,55], a test-collection item $s_i \in \mathcal{T}$ is considered relevant to a query $s_j \in \mathcal{Q}$ if they both belong to the same class.

4.2 Datasets

■ Text retrieval: **AG-news**[3] is text classification and retrieval benchmark [60] consisting in (120,000 train; 7,600 test) sentences, each belonging to exactly one of the 4 classes (sports, world, business, sci/tech). E.g.,"Economic growth in Japan slows down as the country experiences a drop in domestic and corporate spending" (class = business).

■ Image retrieval:

– **Caltech-256** [20] is a benchmark widely used in image retrieval [15] and classification. The data consists of 30,607 images, each of which belongs to exactly one of the 256 categories (e.g., sushi, swan, tripod, etc.).
– **CorelDB** database [51]: consists of 10,800 images, each of which belongs to exactly one of the 80 classes (ship, waterfall, lion, etc.).

■ Word similarity tasks are typically used to evaluate the quality of word embedding models [2,26,31,40,52]. Following [12,52,53], we use five *word similarity* benchmarks, which include three types of similarity ratings: *(i) Semantic similarity*: **SemSim** [44], **Simlex999** [22] and **SimVerb-3500** [19]; *(ii) Relatedness*: **MEN** [3] and **WordSim-353** [18]; *(iii) Visual similarity*: **VisSim** [44] which contains the same data as SemSim, yet word pairs are rated for visual similarity instead of semantic similarity.

[3] https://www.kaggle.com/amananandrai/ag-news-classification-dataset.

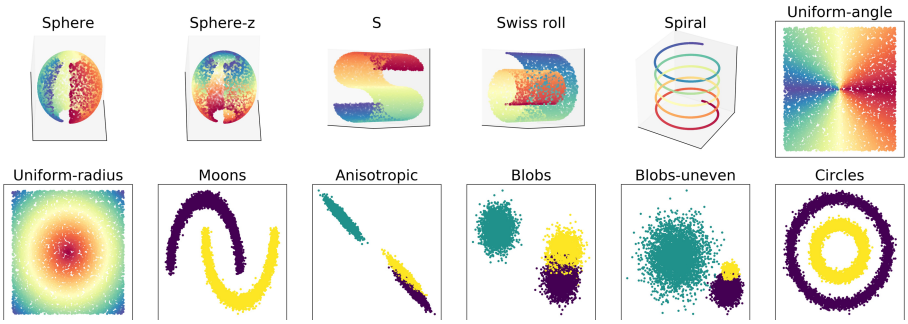

Fig. 2. Synthetic datasets of our laboratory experiment. Color indicates the semantic value of each data point (either a class label or a continues value) – best seen in color. The first five datasets are 3D while the rest are in 2D. In the first seven datasets, the semantic value assigned to data points is continuous, while for the last five datasets the class labels are discrete. (Color figure online)

■ Sentence similarity: Our datasets are from the GLUE[4] and SentEval[5] collections.

- **STS** (Semantic Textual Similarity) [5] is a semantic relatedness benchmark consisting of sentence pairs with a crowd-annotated similarity score. E.g., ("A woman is eating something", "A woman is eating meat") has a score of 3 (out of 5). There are 5,749 train, 1,500 val and 1,379 test pairs.
- **SICK** (Sentences Involving Compositional Knowledge) [36] evaluates compositional distributional semantics. SICK contains sentence pairs along with their semantic relatedness score. E.g., ("Two men are boxing", "Two men are fighting") have a score of 4 (out of 5). SICK has 4,501 train, 501 val and 4,928 test sentence pairs.
- **MSRP** (Microsoft Research Paraphrase Corpus) [17] does not strictly evaluate sentence similarity but paraphrase detection, yet due to functional parallels with the former, we include MSRP in this group. It contains (4,077 train; 1,726 test) sentence pairs along with a label $\{1 = \textit{paraphrase}$ or $0 = \textit{not paraphrase}\}$. MSRP is always used with supervision, thus it may not be the most adequate test-bed for our setting.

■ Visual similarity: **Visual-STS (vis-STS)** [30] is a subset of STS where each textual caption is associated to an image. Here, we only use the images since (a larger super-set of) the sentences are already evaluated in STS. Vis-STS consists of 1,089 images and a single set of 829 image-image pairs along with their ground-truth similarity rating.

[4] https://gluebenchmark.com/tasks.

[5] In contrast to most papers using SICK, MSRP and STS [13,27] we do not use labels. E.g., while [27] learn a logistic regression model to predict the similarity between embedding pairs v_i, v_j, we output the similarity directly (Sect. 4.1).

■ Synthetic data: In contrast to *real-world* tasks, laboratory tasks offer a unique window to study the behavior of feature transforms by having full control of: (i) the (distribution of) feature vectors, (ii) the task itself, i.e., the *assignation of semantic value* to each data point. The majority of our **synthetic (laboratory) datasets** are from sklearn [39], except *sphere-z*, *unif-rad*, *unif-angle* and *spiral* (Fig. 2), which are built by ourselves.

We randomly generate 2,000 train and 200 test data points. Then, we build our *similarity* task by presenting all pairwise combinations of test points to the system, i.e., 40,0000 pairs (= 200 × 200). In the *discrete*-labelled datasets (e.g., *circles*, Fig. 2) where each data point s_i has a class label $l_i \in \{t_1, \cdots, t_C\}$ (where $C = \#$ classes) the ground-truth similarity $y_{i,j} \in \{0, 1\}$ between two points s_i, s_j is 1 if they belong to the same class, or 0 otherwise. In the *continuous*-labelled datasets (e.g., *sphere*), where the assignation of semantic value to each data point is a continuous value $l_i \in \mathbb{R}_+$, the ground truth similarity $y_{i,j}$ between s_i, s_j is the absolute difference: $y_{i,j} = |l_i - l_j| \in \mathbb{R}_+$.

5 Experimental Setup

5.1 Feature Vectors (Embeddings)

We group below our embeddings by the unit that they represent (a word, a sentence or an image). An overview of which embeddings apply to what task can be seen in Table 1.

■ Word-level features:

- **GloVe**[6] [40]: We use 300-*d* vectors pre-trained on the Common Crawl corpus with 840B tokens and a 2.2M-word vocabulary.
- **word2vec (w2v)** [37]: We use the skip-gram 300-*d* embeddings trained on Wikipedia.
- In *word-similarity*, we adopt the publicly available[7] **VGG-128** [6] and **ResNet** [21] visual features from [12]. Notice that unlike the image retrieval and visual-STS tasks, word-similarity datasets do not have any images and hence one needs to find a way to visually represent each word (e.g., 'cat' or 'table') by using external visual data. To this end, [12] used ImageNet [43], and for each image they extracted 128-*d* VGG-128 and 2,048-*d* ResNet features from the last layer (before the softmax) by using the forward pass of the CNN. The final representation for any given word is the average feature vector (centroid) of all available images for this word in ImageNet.

■ Sentence-level features:

- **BERT** [16]: The large uncased version of BERT[8] (24 layers, 1,024 units) is used as a *sentence* feature extractor. We obtain a 1,024-*d* vector from the last

[6] http://nlp.stanford.edu/projects/glove.
[7] http://liir.cs.kuleuven.be/software.html.
[8] Although we are aware that BERT is not meant to represent a single word as it is designed to account for context words, we include BERT in the *word-similarity* tasks for completeness.

layer (24th), before the model top, by average-pooling the output sequence of hidden state vectors, similar to BERT-as-a-service [58]. The model is pre-trained on masked language modeling and next sentence prediction in the Toronto Book Corpus and Wiki.
- **RoBERTa** [33]: We obtain 1,024-d features in an identical manner as in BERT above with the large-version of a case-sensitive RoBERTa model.
- **Skipthoughts** vectors [27] is a popular neural-based universal sentence encoder that learns sentence representations by predicting the surrounding sentences. We use the best-performing 4,800-d vectors (combine-skip) as recommended by the authors.
- **Vector averaging (bag of words):** In the sentence-level tasks (SICK, MSRP, STS and AG-news), we include the baseline sentence representation $v = \frac{1}{m}\sum_{i=1}^{m} v_i$ of averaging word vectors in a sentence $s = (s_1, \cdots, s_m)$, where $v_i = E(s_i)$ and m is the number of words. We add a subscript *avg* to the averaged vectors (e.g., $\text{GloVe}_{\text{avg}}$).

■ Image-level features. Vector dimensionality is in parenthesis: **NASNet** [62] ($d = 4{,}032$), **ResNet-50** [21] ($d = 2{,}048$), **ResNet-inception-v2** [47] ($d = 1{,}536$), **Inception-v3** [48] ($d = 2{,}048$), **VGG19** [45] ($d = 512$), **Xception** [8] ($d = 2{,}048$). In all these CNN networks, the feature vector $v_i = E(s_i)$ for a given image s_i is obtained as the forward pass average-pooled activations from the last layer before the output layer.

5.2 Training Setup and Implementation

- **Given training data:** In **all datasets except word-similarity** (Sect. 4.2), we obtain the training data $V^{tr} = E(S^{tr})$ given in the dataset (yet *without* using class labels). In the case of AG-news, STS, SICK and MSRP we use the provided train-test split (Sect. 4.2). Although CorelDB, Visual-STS and Caltech-256 do not have publicly available train-test set splits, we create the train-test splits ourselves via 3-fold cross-validation. I.e., we split the full data $S = \{v_i\}_{i=1}^{N}$ into 3 disjoint parts and we employ 2 parts for training (S^{tr}) and 1 part for testing (S^{ts}), repeating this 3 times and reporting the average. However, our setting does not require having an available training set. There are two main alternatives to using the given train split: (1) learning $T_\theta()$ in the *test* set; (2) generating S^{tr} ourselves. Although (1) is a legit option (as one does not use labels), it falls within a *transductive learning* setup and assumes a test set of a certain size to enable learning $T_\theta()$. Hence, this is not an option in the case of a singe-instance test set. We also evaluated learning $T_\theta()$ in the test set, and results are discussed in Sect. 6.1.
- **Built training data:** For **word-similarity**, where no training data are available, we use external data to generate V^{tr} (option (2) above). Following [12], we build V^{tr} in *word-similarity* by using features obtained from all words in ImageNet, i.e., visual features for CNNs (ResNet & VGG128) and word embeddings for text (GloVe & word2vec).

Implementation: We use diverse Python libraries, including: Keras [7] for the CNNs, Theano for skipthoughts, sklearn [39], Pytorch and Huggingface [57] for BERT & RoBERTa. We make our code publicly available[9] as well as a Supplement with further specific implementation and hyperparameter details and additional results.

6 Results

Unless otherwise specified, results below are discussed for the *cosine* similarity (Table 1). Performance measures in the tables are according to Sect. 4.1, and scaled × 100, for readability. Table 2 reports **statistical significance** of comparing a given method with the *original* vectors under *cosine* (i.e., the top left corner entry). Each comparison is a two-sided Wilcoxon signed-rank test across the 74 combinations of a *real-world* dataset with an embedding type (i.e., rows in Table 1). We report significance at $p < 0.01$ after a Bonferroni correction for 10 comparisons (7 methods in the first row + Eucl + add. bias + mult. bias)[10]. **Win-tie-loss** results (W, T, L) indicate the number of wins (W), ties (T) and losses (L) of the first method against the second one, across the 74 combinations.

6.1 Real-World Tasks

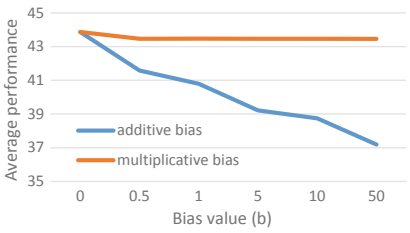

Fig. 3. Averaged results across datasets and features for different values of biases (Sect. 3.2) on *original* vectors. The b=0 point means no bias.

Centeredness: Performance of *original* with an **additive bias** (Sect. 3.2) drastically drops (Fig. 3 and Table 2). This confirms the inadequacy of using uncentered vectors along with the *cosine* similarity. Results of *PCA*, *ctr* and *stz* are unaffected.

• **Centering:** Consistently with the results above, *centering* significantly improved ($p < 10^{-4}$) the *original* features by an absolute 2.5% on average (Table 2), with a win-tie-loss of (W = 52, T = 1, L = 21) (Table 1), hence proving the effectiveness of this method (**RQ 1**).

• **Centeredness of original vectors:** All our CNN vectors (ResNet, etc.) are positive (thus uncentered), and simple statistical inspection reveals that our text vectors are also uncentered. This implies that *centering* has an effect on all our features.

(Non-)homogeneity of variances and scale: In contrast with the large hindering effect of the *additive bias*, performance with the **multiplicative bias**

[9] https://github.com/gcollell/transforms-cosine.

[10] We did not test all pair-wise conditions as our interest is on a specific set of hypotheses.

(Sect. 3.2) barely drops (Fig. 3 and Table 2). This suggests that *centeredness* may have a larger impact on the *cosine* similarity than scale and variance differences across dimensions.

• **Standardizing** is the overall winner in *real-world* tasks (**RQ 1**). It improved significantly ($p < 10^{-6}$) the *orig* features by an absolute 3.3% on average (Table 2) and their win-tie-loss is (W = 60, T = 0, L = 14) (Table 1). Notice that *stz* also centers the vectors.

Cosine versus Euclidean: *Cosine* similarity significantly outperformed ($p < 10^{-6}$) the Euclidean similarity (**RQ 2**) by an average absolute 5.1% (Table 2) and (W = 54, T = 7, L = 13), for the *original* vectors – yet the trend is similar for all transforms. This supports the common practice of defaulting to *cosine* similarity, yet we strongly recommend considering the remarks about *centering* above, to avoid sub-optimal performance. Further, if a labeled validation set is available (e.g., in SICK, STS, or AG-news), one may use it in order to make a more educated choice between *cosine* and *Euclidean* similarity.

Learning times: Remarkably, manifold learning methods are over ×1,000 times slower than *standardizing* (Table 2), and perform markedly worse.

Learning in test set: Notably, *center* and *standardize* can be further improved by learning them in test data (Table 2) – provided the test set is large enough.

Manifold learning methods generally underperform the simple transforms in *real-world* tasks. We emphasize that we do not claim that we fairly portray the full potential of manifold learning methods (and PCA), as we did not tune their hyperparameters (e.g., dimensionality) with a validation set for the sake of comparability with the simple transforms – as our setting does not assume a validation set.

PCA improved *orig* features by 1.8% on average (Table 2) and (W = 53, T = 0, L = 21).

Failure cases: Notably, VGG19 was not improved by any method in any dataset (Table 1), and all methods fared poorly in MSRP. However, the performance loss by *standardizing* or *centering* is small in MSRP, which suggests that, in the absence of a validation set for making more informed decisions, the large upside of defaulting to standardizing may offset its eventual and rather small potential performance downside.

Consistency: Some methods that perform poorly on average such as *Iso* or *wht* (Table 2) eventually hit the most spectacular gains (and losses) (Table 2). This contrasts with *stz* and *ctr* which tend to have less "volatility" and exhibit more consistent gains.

Table 1. Results with *cosine* similarity on **real-world** tasks. Since performance trends are similar, the *word-similarity* table (left) includes only the visual subsets, i.e., word-pairs for which images are available for both words – number of instances is in parenthesis. Results in all sets are in the Supplement. Best-performing method per row is boldfaced.

		orig	ctr	stz	wth	iso	LLE	PCA
wordsim (63)	GloVe	63.2	61.3	67.6	**69.6**	43.9	50.1	59.4
	w2v	**66.9**	64.9	65.5	62	58.8	25.3	63
	BERT	20.8	30.1	28	**47.4**	16.7	26.3	29.9
	RoBERTa	23.9	23.5	26.7	**44.9**	23.1	11.2	22.4
	ResNet	42.3	48.9	48.1	28.7	**51.7**	43.7	48.6
	VGG128	44.8	49.2	49.1	50	**55.4**	54	46.7
men (795)	GloVe	80.1	80	**82.7**	79.9	76.6	67	80.2
	w2v	78.7	81	**81.1**	74.3	70.8	53.5	80.9
	BERT	32.7	31.9	35.9	**49.5**	25.3	24.1	31.3
	RoBERTa	20.9	29.5	32.2	**48**	25.1	20.7	27.6
	ResNet	56.7	59	60.7	36.8	**60.9**	42.3	59.2
	VGG128	59.3	58.9	59.8	54	**60**	42.9	58.8
semSim (5,238)	GloVe	76.8	74.6	**78**	62	77.3	61.2	75.1
	w2v	74.2	77.3	77.3	54.1	71.2	50	**77.6**
	BERT	23.4	22.6	25.2	**28.3**	17.3	22.7	22.8
	RoBERTa	20	28.5	**30.3**	26.1	26.2	22.8	28.2
	ResNet	53.4	67.6	67.6	11.7	**70.3**	39.1	67.8
	VGG128	53.4	65.8	65.1	37.9	**69**	36	66.1
visSim (5,238)	GloVe	60.6	60.6	**62.9**	53.7	61.5	47.7	61
	w2v	57.6	60.8	60.8	47.7	54.8	37.9	**61**
	BERT	16.2	16.7	18.4	**23.7**	12	14.8	16.7
	RoBERTa	15.7	21.1	**22.4**	22.2	18.6	15.4	20.7
	ResNet	54.3	60.6	**61.7**	14.5	57.9	37	60.8
	VGG128	56	60.7	**61.2**	42.9	60	35.2	60.7
simlex (261)	GloVe	37.1	36.1	42	45.1	35.6	**45.3**	35.7
	w2v	43.5	44.3	**44.9**	41.7	41.9	35.2	43.8
	BERT	24.3	21.6	23.7	**36.6**	18.5	16	20.9
	RoBERTa	−7.6	−6.6	−4.5	**19.3**	−11	−3.8	−7.7
	ResNet	40.9	45	45.6	36	**47.3**	38.8	45.5
	VGG128	40.6	42.6	42.2	40.3	43	34	**43.3**
SimVerb (41)	GloVe	32	29.8	**34.3**	22.8	10.5	−6.6	34.1
	w2v	30.8	19.7	21.3	**31.3**	−4.9	−3.1	12.7
	BERT	−7.2	−8.2	−6.6	13.6	−11.6	**21.5**	−7.1
	RoBERTa	4.7	1.5	2	−6.1	−9.8	1.5	**5.7**
	ResNet	21.1	21.2	**27.7**	22	23.7	4.5	19.1
	VGG128	23.5	23.4	20.6	**52**	20.8	14.6	22

		orig	ctr	stz	wht	iso	LLE	PCA
STS	$\text{GloVe}_{\text{avg}}$	50	57.6	58.9	**68.1**	37.7	42.5	51.2
	w2v_{avg}	55.1	59.6	60.1	**66**	38.2	42.8	56.6
	skipthoug	34.5	38.8	44.9	**59.3**	17.2	30.9	35.3
	BERT	47.7	53.5	55.1	**64.5**	34.4	42.3	51.1
	RoBERTa	44.9	58.6	62.7	**69.1**	38.6	42.8	55.1
SICK	$\text{GloVe}_{\text{avg}}$	56	58.5	**59.3**	58	51.9	45.6	55.2
	w2v_{avg}	58.5	60	**60.3**	56.3	51.6	46.1	57.9
	skipthoug	56.5	**58.5**	56.1	43.3	48.4	44	57.1
	BERT	53.8	57.8	58.2	**58.4**	50.7	44.5	55.8
	RoBERTa	58.1	61.9	**63.8**	61.5	54.4	46.4	59.9
MSRP	$\text{GloVe}_{\text{avg}}$	**39.9**	37.2	38	39.5	16.3	16.1	34.4
	w2v_{avg}	**38.8**	35.6	36.2	37.8	14	21.6	33.7
	skipthoug	15.3	19.6	21.1	**32.9**	7.6	8.8	16.4
	BERT	31.5	30	30.8	**38.2**	11.6	15.3	26.3
	RoBERTa	**43.7**	39.5	39.5	42.3	10.9	15.8	36.6
AG-news	$\text{GloVe}_{\text{avg}}$	48.5	55.6	52.6	30.6	**62.9**	59.4	55.7
	w2v_{avg}	54.9	60.9	60	31.3	**66.5**	56.7	61.2
	skipthoug	36.6	**39.3**	38.5	27.1	**39.3**	37.8	**39.3**
	BERT	49.9	57.6	56.9	29.6	**66.2**	50.5	57.6
	RoBERTa	47.3	55	54.8	29.6	**67.8**	54.1	55.1
vis-STS	Incptn-v3	44.9	**55.3**	54.6	10.3	53.4	37.8	54.2
	ResNet-in	45.2	56.1	55.5	9.5	**57.9**	37.2	54.7
	ResNet	61.7	**64.6**	60.9	10.8	59.2	43.2	63.5
	xception	56.2	**59.2**	59.1	14.3	47.9	38	58.2
	VGG19	**63.4**	56.5	53.2	5.8	53.2	32.4	54.2
	NASNet	49.9	**58.7**	57.4	15.5	53.7	34.7	57.2
Caltech	Incptn-v3	45.8	47.9	**48.5**	27.4	42	45.5	48.3
	ResNet-in	54.9	55	55	38.7	55.2	52.8	**55.4**
	ResNet	40.6	41.1	**42.7**	22.5	36.2	38.9	41.3
	xception	46.8	50	49.9	27.8	41.7	46.5	**51**
	VGG19	**35.1**	34.3	35.2	25.8	26.6	26.7	33.8
	NASNet	60.4	60.6	60.2	29.9	58.3	57.2	**61.6**
CorelDB	Incptn-v3	43.8	46.9	47.3	11.6	42.6	**50.2**	47.9
	ResNet-in	52.3	53.5	53.2	18.6	**55.4**	54.5	55.1
	ResNet	45.7	47.2	46.1	12	**48.5**	52.4	48.1
	xception	47.4	50.4	49.7	13.6	47	**52.8**	51.9
	VGG19	36.9	38.2	37.7	18	34.8	**38.8**	38.5
	NASNet	57.9	59.2	57.4	2.2	56.6	57.7	**60.9**

6.2 Synthetic Data

Unlike *real-world* data (Sect. 6.1) where vectors and semantic value assignment (i.e., the task) cannot be visualized, synthetic data enable intuitively grasping and visualizing the effect that transforming vectors (**RQ 1**) have on the similarity measures (**RQ 2**).

Table 2. Averaged results across *real-world* datasets and features. Rows include (in order) results of: (i) *cosine* similarity (i.e., averaged results of Tab. 1); (ii) *Euclidean* similarity; (iii) *additive* and (iv) *multiplicative* bias (Sect. 3.2) ($b = 10$); (v) learning T_θ in the test set, and (vi) *training times* (in seconds). Despite omitting datasets, this table portrays a representative summary of the performance landscape. SDs are omitted for being uninformative, as they reflect inter-dataset variance. For individual results, see Table 1, the Supplement and win-tie-loss mentions in the text. Asterisks (*) indicate statistically different performance ($p < 0.01$) from *orig* × cos (two-sided).

	orig	ctr	stz	wht	nrm	iso	LLE	PCA
cos	43.9	46.4*	47.2*	36.2*	43.9	40.9	35.7*	45.7*
Eucl	38.8*	38.8	38.1	21.6	43.9	35.7	23.1	39.8
add. bias	38.7*	46.4	47.2	35.1	38.7	40.9	35.5	45.7
mult. bias	43.5	45.8	47.2	36.1	43.5	40.6	35.8	44.9
learn in test	43.9	47.3	47.9	35.8	43.9	44.5	38.1	46.8
train time	0	0.01	0.3	7.9	2.1e-05	1174.5	965.7	2.4

Centeredness: Crucially, *original* vectors in synthetic tasks are generally centered while in real-world tasks features are uncentered (Sect. 6.1). It is reasonable to not expect that features will be natively centered at $\vec{0}$, unless explicitly imposed. Thus, using uncentered vectors *orig (add)* as a reference point in Table 3 may be more "realistic" than *orig*.

- Applying an **additive bias** (*orig (add)*) generally hinders the *original* vectors (with *cosine*) (Table 3), yet one can find a pathological case in *circles*, where having centered vectors (e.g., *orig* or *ctr*) is detrimental. The reason being that, with centered vectors, the $\vec{0}$ point falls inside the circles (Fig. 2), hence the angle (or *cosine* similarity) which stems from $\vec{0}$, is utterly unhelpful to tell apart the inner from the outer circle. Although it is important to gain insight on these cases with synthetic data, real-world feature vectors (and tasks) are unlikely to exhibit this onion-like structure unless explicitly imposed [38, 59]. Thus, there is no substitute for a systematic study in *real world* tasks (Table 1).

Task versus vectors: A key question that this paper answers is whether it suffices to look at (the statistics of) the vectors alone in order to tell when a transform will perform well. *Unif-radius* and *unif-angle* illustrate a negative answer. *All* methods fail at *unif-radius* (radius matters) while they all do reasonably well in *unif-angle* (angle matters). The only difference is the assignment of a semantic value to data points, i.e., the *task* itself. Thus, vectors alone do not suffice to determine effectiveness of a transform but they must be considered along with the task. Many *real-world* instances support this conclusion, e.g., *stz* improving NASNet in vis-STS, yet not in Caltech nor in CorelDB.

Failure cases: *Circles* illustrates a task where *cosine* similarity is entirely unhelpful to tell both classes apart (and the Euclidean only barely useful) for the

Table 3. Results on **synthetic** datasets. The (add) and (mult) indicate that an additive or multiplicative bias, respectively, is added to the method (Sect. 3.2). SDs are left to the Supplement.

	orig		orig (add)		orig (mult)		ctr		stz		wth		nrm		iso		LLE		PCA	
	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl	Cos	Eucl
sphere	65.1	65.1	44	65.1	58.5	54.9	62.1	65.1	62.5	65.8	65.4	65.3	65.1	65.1	59.7	76.7	58.2	71.3	50.7	50.2
sphere-z	53.1	53.1	40.6	53.1	46.5	45.4	53.5	53.1	51.6	49.7	51.4	50.1	53.1	53.1	52.7	49.7	51.1	44	59.6	64.2
s	82.7	88.3	74.2	88.3	70.3	72.8	76.3	88.3	53.3	58.9	66.2	74.3	82.7	82.7	76.3	98.7	70.5	77.5	75.3	91.3
roll	20.2	23.9	19.6	23.9	17.9	21.7	22.1	23.9	21.2	22.1	24.5	28.9	20.2	20.2	75.3	96	60.8	71.1	23.8	31.8
spiral	40.3	100	89.4	100	36.9	94.3	83.2	100	41.5	55.5	13.7	21.4	40.3	40.3	82.8	97.2	71.5	76.1	79.1	100
unif-ang	62.1	50.8	28.2	50.8	55.7	44.2	62	50.8	62	50.8	62.1	50.8	62.1	62.1	62	50.9	61.8	49.8	62	50.8
unif-rad	0	5.2	3.5	5.2	−0.1	7.7	0	5.2	0	5.2	0	5.2	0	0	0	5.1	0	6	0	5.2
moons	43	41.2	49.7	41.2	43.3	42.6	38.4	41.2	47.6	52.4	47	51.6	43	43	0	16.9	32.4	53.7	38.4	41.2
Aniso	54.3	62.1	54.3	62.1	52.8	61.1	62.1	62.1	61.5	61.1	58.2	62	54.3	54.3	39.9	40.3	53.7	63.1	62.1	62.1
blobs	57.9	66.3	53.4	66.3	56	64.3	64.2	66.3	64.8	66.4	58.2	62	57.9	57.9	39	42.6	55.8	66.1	64.2	66.3
blobs-un	54.8	61.2	48.1	61.2	52.4	59.1	63.8	61.2	64	60.4	55	55.1	54.8	54.8	58.9	42.3	55.9	60.2	63.8	61.2
circles	−0.1	13.4	11.8	13.4	0	13.1	−0.1	13.4	−0.1	13.4	−0.1	13.4	−0.1	−0.1	40.4	28.4	35.2	53.6	−0.1	13.4

regular methods, yet manifold learning methods fare better (Table 3). Further notice the detrimental effect of *normalizing* with the Euclidean similarity in the same dataset, as normalizing collapses both circles into one. We also highlight the general failure of *all* methods in our own "stress test" task, *unif-rad*. Likely, polar coordinates would have done a better job.

7 Conclusions and Future Work

Limitations. The answer to whether any of our top-performing transforms is a universal recipe to improve (text or image) features, is a negative one. As usual, there is *no free lunch*. However, this study strives to include a representative and reasonable number of datasets and varied tasks to gain insight on the success rate and effect size of each transform. **Performance trends** showcase promising potential on defaulting to *center*, *PCA* or *standardize* the features in applications, as well as using *cosine*-based (instead of Euclidean) semantic match. That said, our task selection is not exhaustive and hence we encourage researchers to report results on new tasks and datasets.

A word of caution. In line with [33] and [32], an important contribution of this work is rising awareness about the potential source of improvements in some word and sentence embedding models, which are often tested in semantic-similarity tasks and default to *cosine* similarity. As shown, feature re-scaling can have a much greater impact on the overall performance than the embedding model itself. Hence, it is crucial to control for any possible feature re-scalings occurring in any step of the pipeline.

Acknowledgment. This research was supported by the ERC Advanced Grant CALCULUS (H2020-ERC-2017-ADG 788506).

References

1. Artetxe, M., Labaka, G., Agirre, E.: Generalizing and improving bilingual word embedding mappings with a multi-step framework of linear transformations. In: AAAI, pp. 5012–5019 (2018)
2. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: ACL, pp. 238–247 (2014)
3. Bruni, E., Tran, N.K., Baroni, M.: Multimodal distributional semantics. J. Artif. Intell. Res. **49**, 1–47 (2014)
4. Cao, X.H., Stojkovic, I., Obradovic, Z.: A robust data scaling algorithm to improve classification accuracies in biomedical data. BMC Bioinformatics **17**(1), 359 (2016)
5. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055 (2017)
6. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. In: BMVC (2014)
7. Chollet, F., et al.: Keras (2015). https://github.com/keras-team/keras
8. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: CVPR, pp. 1251–1258 (2017)
9. Cinbis, R.G., Verbeek, J., Schmid, C.: Unsupervised metric learning for face identification in TV video. In: ICCV, pp. 1559–1566. IEEE (2011)
10. Coifman, R.R., Lafon, S.: Diffusion maps. Appl. Comput. Harmonic Anal. **21**(1), 5–30 (2006)
11. Collell, G., Moens, M.F.: Do neural network cross-modal mappings really bridge modalities? In: ACL, pp. 462–468 (2018)
12. Collell, G., Zhang, T., Moens, M.F.: Imagined visual representations as multimodal embeddings. In: AAAI, pp. 4378–4384. AAAI (2017)
13. Conneau, A., Kiela, D.: Senteval: An evaluation toolkit for universal sentence representations. arXiv preprint arXiv:1803.05449 (2018)
14. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML, pp. 209–216. Corvalis, Oregon, USA (June 2007)
15. Deng, J., Berg, A.C., Fei-Fei, L.: Hierarchical semantic indexing for large scale image retrieval. In: CVPR, pp. 785–792. IEEE (2011)
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp. 4171–4186 (2019)
17. Dolan, B., Quirk, C., Brockett, C.: Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In: COLING, pp. 350–356 (2004)
18. Finkelstein, L., et al.: Placing search in context: the concept revisited. In: WWW, pp. 406–414. ACM (2001)
19. Gerz, D., Vulić, I., Hill, F., Reichart, R., Korhonen, A.: Simverb-3500: a large-scale evaluation set of verb similarity. In: EMNLP, pp. 2173–2182 (2016)
20. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
22. Hill, F., Reichart, R., Korhonen, A.: Simlex-999: evaluating semantic models with (genuine) similarity estimation. Comput. Linguist. **41**(4), 665–695 (2015)

23. Jiang, J., Wang, B., Tu, Z.: Unsupervised metric learning by self-smoothing operator. In: ICCV, pp. 794–801. IEEE (2011)
24. Jones, W.P., Furnas, G.W.: Pictures of relevance: a geometric analysis of similarity measures. J. Am. Soc. Inform. Sci. **38**(6), 420–442 (1987)
25. Kessy, A., Lewin, A., Strimmer, K.: Optimal whitening and decorrelation. Am. Stat. **72**(4), 309–314 (2018)
26. Kiela, D., Bottou, L.: Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In: EMNLP, pp. 36–45 (2014)
27. Kiros, R., et al.: Skip-thought vectors. In: NIPS, pp. 3294–3302 (2015)
28. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
29. Kruskal, J.B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika **29**(1), 1–27 (1964)
30. de Lacalle, O.L., Soroa, A., Agirre, E.: Evaluating multimodal representations on sentence similarity: vSTS, visual semantic textual similarity dataset. arXiv preprint arXiv:1809.03695 (2018)
31. Lazaridou, A., Baroni, M., et al.: Combining language and vision with a multimodal skip-gram model. In: NAACL, pp. 153–163 (2015)
32. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Trans. Assoc. Comput. Linguist. **3**, 211–225 (2015)
33. Liu, Y., et al.: Roberta: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
34. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(Nov), 2579–2605 (2008)
35. Manning, C.D., Schütze, H., Raghavan, P.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
36. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R., et al.: A sick cure for the evaluation of compositional distributional semantic models. In: LREC, pp. 216–223 (2014)
37. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
38. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. In: NIPS, pp. 6338–6347 (2017)
39. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
40. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
41. Raghavan, V.V., Wong, S.M.: A critical analysis of vector space model for information retrieval. J. Am. Soc. Inf. Sci. **37**(5), 279–287 (1986)
42. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
43. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
44. Silberer, C., Lapata, M.: Learning grounded meaning representations with autoencoders. In: ACL, pp. 721–732 (2014)
45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
46. Socher, R., Ganjoo, M., Manning, C.D., Ng, A.: Zero-shot learning through cross-modal transfer. In: NIPS, pp. 935–943 (2013)

47. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-ResNet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261 (2016)
48. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR, pp. 2818–2826 (2016)
49. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
50. Wang, B., Yang, Y., Xu, X., Hanjalic, A., Shen, H.T.: Adversarial cross-modal retrieval. In: ACM Multimedia, pp. 154–162 (2017)
51. Wang, J.Z., Li, J., Wiederhold, G.: Simplicity: semantics-sensitive integrated matching for picture libraries. IEEE Trans. Pattern Anal. Mach. Intell. **23**(9), 947–963 (2001)
52. Wang, S., Zhang, J., Zong, C.: Associative multichannel autoencoder for multimodal word representation. In: EMNLP, pp. 115–124 (2018)
53. Wang, S., Zhang, J., Zong, C.: Learning multimodal word representation via dynamic fusion methods. In: AAAI (2018)
54. Wang, W., Ooi, B.C., Yang, X., Zhang, D., Zhuang, Y.: Effective multi-modal retrieval based on stacked auto-encoders. Proc. VLDB Endow. **7**(8), 649–660 (2014)
55. Wei, Y., et al.: Cross-modal retrieval with CNN visual features: a new baseline. IEEE Trans. Cybern. **47**(2), 449–460 (2016)
56. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**(2), 207–244 (2009)
57. Wolf, T., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv abs/1910.03771 (2019)
58. Xiao, H.: bert-as-service (2018). https://github.com/hanxiao/bert-as-service
59. Xing, C., Wang, D., Liu, C., Lin, Y.: Normalized word embedding and orthogonal transform for bilingual word translation. In: ACL, pp. 1006–1011 (2015)
60. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: NIPS, pp. 649–657 (2015)
61. Zhang, Y., Gong, B., Shah, M.: Fast zero-shot image tagging. In: CVPR, pp. 5985–5994. IEEE (2016)
62. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR, pp. 8697–8710 (2018)