Knowledge and Information Systems (2021) 63:565–588
https://doi.org/10.1007/s10115-020-01531-7

REGULAR PAPER

Closed form word embedding alignment

Sunipa Dev[1] · **Safia Hassan**[1] · **Jeff M. Phillips**[1]

Received: 28 January 2020 / Revised: 12 November 2020 / Accepted: 15 November 2020 /
Published online: 9 January 2021

Abstract
We develop a family of techniques to align word embeddings which are derived from different source datasets or created using different mechanisms (e.g., GloVe or word2vec). Our methods are simple and have a closed form to optimally rotate, translate, and scale to minimize root mean squared errors or maximize the average cosine similarity between two embeddings of the same vocabulary into the same dimensional space. Our methods extend approaches known as absolute orientation, which are popular for aligning objects in three dimensions, and generalize an approach by Smith et al. (ICLR 2017). We prove new results for optimal scaling and for maximizing cosine similarity. Then, we demonstrate how to evaluate the similarity of embeddings from different sources or mechanisms, and that certain properties like synonyms and analogies are preserved across the embeddings and can be enhanced by simply aligning and averaging ensembles of embeddings.

Keywords Word embeddings · GloVe · Word2Vec · ELMo

1 Introduction

Embedding complex data objects into a high-dimensional, but easy to work with, feature space has been a popular paradigm in data mining and machine learning for more than a decade [32,33,39,42]. This has been especially prevalent recently as a tool to understand language, with the popularization through word2vec [25,27] and GloVe [29]. These approaches take as input a large corpus of text, and map each word which appears in the text to a vector representation in a high-dimensional space (typically $d = 300$ dimensions).

These word vector representations began as attempts to estimate similarity between words based on the context of their nearby text, or to predict the likelihood of seeing words in the context of another. Other more powerful properties were discovered. Consider each word gets mapped to a vector $v_{\text{word}} \in \mathbb{R}^d$.

✉ Sunipa Dev
sunipad@cs.utah.edu

Safia Hassan
safiahassan609@gmail.com

Jeff M. Phillips
jeffp@cs.utah.edu

[1] School of Computing, Salt Lake City, USA

- *Synonym similarity:* Two synonyms (e.g., v_{car} and $v_{\text{automobile}}$) tend to have small Euclidean distances and large inner products, and are often nearest neighbors.
- *Linear relationships:* For instance, the vector subtraction between countries and capitals (e.g., $v_{\text{Spain}} - v_{\text{Madrid}}$, $v_{\text{France}} - v_{\text{Paris}}$, $v_{\text{Germany}} - v_{\text{Berlin}}$) is similar. Similar vectors encode gender (e.g., $v_{\text{man}} - v_{\text{woman}}$), tense ($v_{\text{eat}} - v_{\text{ate}}$), and degree ($v_{\text{big}} - v_{\text{bigger}}$).
- *Analogies:* The above linear relationships could be transferred from one setting to another. For instance the gender vector $v_{\text{man}} - v_{\text{woman}}$ (going from a female object to a male object) can be transferred to another more specific female object, say v_{queen}. Then, the result of this vector operation is $v_{\text{queen}} + (v_{\text{man}} - v_{\text{woman}})$ is close to the vector v_{king} for the word "king." This provides a mechanism to answer analogy questions such as "woman:man::queen:?"
- *Classification:* More classically [32,33,39,42], one can build linear classifiers or regressors to quantify or identify properties like sentiment.

At least in the case of GloVe, these linear substructures are not accidental; the embedding aims to preserve inner product relationships. Moreover, these properties all enforce the idea that these embeddings are useful to think of inheriting a Euclidean structure, i.e., it is safe to represent them in $\mathbb{R}^d$ and use Euclidean distance.

However, there is nothing extrinsic about any of these properties. A rotation or scaling of the entire dataset will not affect synonyms (nearest neighbors), linear substructures (dot products), analogies, or linear classifiers. A translation will not affect distance, analogies, or classifiers, but will affect inner products since it effectively changes the origin. These substructures (i.e., metric balls, vectors, halfspaces) can be transformed in unison with the embedded data. Indeed Euclidean distance is the only metric on d-dimensional vectors that is rotation invariant.

The intrinsic nature of these embeddings and their properties adds flexibility that can also be a hinderance. In particular, we can embed the same dataset into $\mathbb{R}^d$ using two approaches, and these structures cannot be used across datasets. Or two datasets can both be embedded into $\mathbb{R}^d$ by the same embedding mechanism, but again the substructures do not transfer over. That is, the same notions of similarity or linear substructures may live in both embeddings, but have different meanings with respect to the coordinates and geometry. This makes it difficult to compare approaches; the typical way is to just measure a series of accuracy scores, for instance in recovering synonyms [21,27]. However, these single performance scores do not allow deeper structural comparisons.

Another issue is that it becomes challenging (or at least messier) to build ensemble structures for embeddings. For instance, some groups have built word vector embeddings for enormous datasets (e.g., GloVe embedding using 840 billion tokens from Common Crawl, or the word2vec embedding using 100 billion tokens of Google News), which costs at least tens of thousands of dollars in cloud processing time. Given several such embeddings, how can these be combined to build a new single better embedding without revisiting that processing expense? How can a new (say specialized) dataset from a different domain use a larger high-accuracy embedding?

Our approach and results In this paper, we provide a simple closed-form method to optimally align two embeddings. These methods find optimal rotation (technically an orthogonal transformation) of one dataset onto another, and can also solve for the optimal scaling and translation. They are optimal in the sense that they minimize the sum of squared errors under the natural Euclidean distance between all pairs of common data points, or they can maximize the average cosine similarity.

The methods we consider are easy to implement, and are based on 3-dimensional shape alignment techniques common in robotics and computer vision called "absolute orientation." We observe that these approaches extend to arbitrary dimensions d; the same solution for the optimal orthogonal transformation was also recently re-derived by Smith et al. [40].

In this paper, we also show that an approach to choose the optimal scaling of one dataset onto another [18] does not affect the optimal choice of rotation. Hence, the choice of translation, rotation, and scaling can all be derived with simple closed-form operations.

We then apply these methods to align various types of word embeddings, providing new ways to compare, translate, and build ensembles of them. We start by aligning datasets to themselves with various types of understandable noise; this provides a method to calibrate the error scores reported in other settings. We also demonstrate how these aligned embeddings perform on various synonym and analogy tests, whereas without alignment the performance is very poor. The results with scaling, translation, and weighting all consistently improve upon the results for only rotation as advocated by Smith et al. [40].

Moreover, we show that we can boost embeddings, showing improved results when aligning various embeddings, and taking simple averages of the embedded words from different datasets. The results from these boosted embeddings provide the best known results for various analogy and synonym tests. More extensive use of ensembles should be possible, and it could be applied to a wider variety of data types where Euclidean feature embeddings are known, such as for graphs [6,9,13,14,30], images [2,22], and for kernel methods [32,33].

This alignment can also aid translation, wherein an alignment learned from a small set of words whose translation is known, we can obtain an alignment of a much larger set of words. We also show how aligning two low-resource languages independently to a well-documented and accurate intermediate language can aid in translation between the first two languages.

Finally, in the last few years, contextualized embeddings, such as BERT [8] and ELMo [31], which embed a word differently each time, based on the context it appears in, have become increasingly pervasively used in language processing tasks such as textual entailment and co-reference resolution. We show that a simple average of the contexts allows our techniques to efficiently extend to modeling a more complex multi-way alignment among word representations.

1.1 Word embedding mechanisms

There are several different mechanisms today to embed words into a high-dimensional vector space. They can primarily be divided into mechanisms: first, those that produce a single vector for each word (e.g., GloVe [29]), thus leading to a dictionary-like structure, and the second (ELMO [31], BERT [8]) producing a function instead of a vector for a word such that given a context, the vector for the word is generated. This implies that, different word senses (river *bank* versus financial institution *bank*) lead to differently embedded representations of the same word. Our experiments primarily examine the first kind as vector distances are interpretable there.

The word embedding mechanisms we use here are:

RAW: has as many dimensions as there are words, each dimension corresponds with the rate of co-occurrences with a particular word; it is in some sense what other sophisticated models such as GloVe are trying to understand and approximate at much lower dimensions.

GloVe: uses an unsupervised learning algorithm [29] for obtaining vector representations for words based on their aggregated global word–word co-occurrence statistics from a corpus.

word2vec: builds representations of words so the cosine similarity of their embeddings can be used to predict a word that would fit in a given context and can be used to predict the context that would be an appropriate fit for a given word.
FastText: scales [19] these methods of deriving word representations to be more usable with larger data with less time cost. The also include sub-word information [4] to be able to generate word embeddings for words unseen during training. We use FastText word representations for Spanish and French from the library provided (https://fasttext.cc/docs/en/crawl-vectors.html) for our translation experiments in Sect. 4.2.
More recently, contextual embeddings which produce vectors for word in every distinct context have become popular, such as ELMO [31] and BERT [8]. These embeddings differ from the other embeddings mentioned above in that they do not give a look-up table/dictionary in the end wherein each word has one exact corresponding vector representation. We describe an extension of our method of absolute orientation for alignment for these embeddings in Sect. 2.1.

2 Closed-form point set alignment: classic and new results

In many classic computer vision and shape analysis problems, a common problem is the alignment of two (often 3-dimensional) shapes. The most clean form of this problem starts with two points sets $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$, each of size n, where each a_i corresponds with b_i (for all $i \in 1, 2, \ldots, n$). Generically, we can say each $a_i, b_i \in \mathbb{R}^d$ (without restricting d), but as mentioned the focus of this work was typically restricted to $d = 3$ or $d = 2$. Then, the standard goal was to find a rigid transformation—a translation $t \in \mathbb{R}^d$ and rotation $R \in \mathbb{SO}(d)$—to minimize the root mean squared error (RMSE). An equivalent formulation is to solve for the sum of squared errors as

$$(R^*, t^*) = \underset{t \in \mathbb{R}^d, R \in \mathbb{SO}(d)}{\operatorname{argmin}} \sum_{i=1}^{n} \|a_i - (b_i R + t)\|^2. \tag{1}$$

For instance, this is one of the two critical steps in the well-known iterative closest point (ICP) algorithm [3,7].

In the 1980s, several *closed-form* solutions to this problem were discovered; their solutions were referred to as solving *absolute orientation*. The most famous paper by Horn [18] uses unit quaternions. However, this approach seems to have been known earlier [11], and other techniques using rotation matrices and the SVD [1,15], rotation matrices and an eigen-decomposition [37,38], and dual number quaternions [41], have also been discovered. In 2 or 3 dimensions, all of these approaches take linear (i.e., $O(n)$) time, and in practice have roughly the same run time [10].

In this document, we focus on the singular value decomposition (SVD)-based approach of Hanson and Norris [15], since it is clear, has an easy analysis, and unlike the quaternion-based approaches which only work for $d = 3$, generalizes to any dimension d. A singular value decomposition (SVD) factorizes a matrix of dimensions $m \times n$ to produce two orthonormal matrices (U and V) and a diagonal matrix (S) to satisfy the linear transformation $x = Ax$. The orthonormal matrices capture the rotation or reflection of the space while the diagonal matrix S captures the singular values which interpret the magnitude of information along each of the respective dimensions. Hanson and Norris's approach decouple the rotation from the translation and solve for each independently. It further uses the orthonormal matrices produced by SVD to determine the rotation. In particular, this approach first finds the means

$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$ and $\bar{b} = \frac{1}{n}\sum_{i=1}^{n} b_i$ of each dataset. Then, it creates centered versions of those datasets $\hat{A} \leftarrow (A, \bar{a})$ and $\hat{B} \leftarrow (B, \bar{b})$. Next we need to compute the RMSE-minimizing rotation (all rotations are then considered around the origin) on centered data sets $\hat{A}$ and $\hat{B}$. First compute the sum of outer products $H = \sum_{i=1}^{n} \hat{b}_i^T \hat{a}_i$, which is a $d \times d$ matrix. We emphasize $\hat{a}_i$ and $\hat{b}_i$ are row vectors, so this is an outer product, not an inner product. Next take the singular value decomposition of H so $[U, S, V^T] = \mathsf{svd}(H)$, and the ultimate rotation is $R = UV^T$. We can create the rotated version of B as $\tilde{B} = \hat{B}R$ so we rotate each point as $\tilde{b}_i = \hat{b}_i R$.

Within this paper, we will use this approach, as outlined in Algorithm 1, to align several datasets each of which have no explicit intrinsic properties tied to their choice of rotation. We in general do not use the translation step for two reasons. First, this effectively changes the origin and hence the inner products. Second, we observe the effect of translation is usually small, and typically does not improve performance.

Algorithm 1 AO- ROTATION(A, B)

Compute the sum of outer products $H = \sum_{i=1}^{n} b_i^T a_i$
Decompose $[U, S, V^T] = \mathsf{svd}(H)$
Build rotation $R = UV^T$
return $\tilde{B} = BR$ so each $\tilde{b}_i = b_i R$

Technically, this may allow R to include mirror flips, in addition to rotations. These can be detected (if the last singular value is negative) and factored out by multiplying by a near-identity matrix $R = UI_{-}V^T$ where I_{-} is identity, except the last term is changed to -1. We ignore this issue in this paper, and henceforth consider orthogonal matrices $R \in \mathbb{O}(d)$ (which includes mirror flips) instead of just rotations $R \in \mathbb{SO}(d)$. For simpler nomenclature, we still refer to R as a "rotation."

We discuss here a few other variants of this algorithm which take into account translation and scaling between A and B.

Algorithm 2 ABSOLUTEORIENTATION(A, B) [15]

Compute $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$ and $\bar{b} = \frac{1}{n}\sum_{i=1}^{n} b_i$
Center $\hat{A} \leftarrow (A, \bar{a})$ so each $\hat{a}_i = a_i - \bar{a}$, and similarly $\hat{B} \leftarrow (B, \bar{b})$
Compute the sum of outer products $H = \sum_{i=1}^{n} \hat{b}_i^T \hat{a}_i$
Decompose $[U, S, V^T] = \mathsf{svd}(H)$
Build rotation $R = UV^T$
Rotate $\tilde{B} = \hat{B}R$ so each $\tilde{b}_i = \hat{b}_i R$
Translate $B^* \leftarrow (\tilde{B}, -\bar{a})$ so each $b_i^* = \tilde{b}_i + \bar{a}$
return B^*

Note that the rotation R and translation $t = -\bar{b} + \bar{a}$ derived within this Algorithm 2 are not exactly the optimal (R^*, t^*) desired in formulation (1). This is because the order these are applied, and the point that the dataset is rotated around is different. In formulation (1), the rotation is about the origin, but the dataset is not centered there, as it is in Algorithm 2.

Translations To compare with the use of also optimizing for the choice of translations in the transformation, we formally describe this procedure here. In particular, we can decouple rotations and translations, so to clarify the discrepancy between Algorithm 2 and Eq. (1),

we use a modified version of the above procedure. In particular, we first center all datasets, $\hat{A} \leftarrow A$ and $\hat{B} \leftarrow B$, and henceforth can know that they are already aligned by the optimal translation. Then, once they are both centered, we can then call AO- ROTATION($\hat{A}$, $\hat{B}$). This is written explicitly and self-contained in Algorithm 3.

Algorithm 3 AO- CENTERED(A, B)

Compute $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$ and $\bar{b} = \frac{1}{n}\sum_{i=1}^{n} b_i$
Center $\hat{A} \leftarrow (A, \bar{a})$ so each $\hat{a}_i = a_i - \bar{a}$, and similarly $\hat{B} \leftarrow (B, \bar{b})$
Compute the sum of outer products $H = \sum_{i=1}^{n} \hat{b}_i^T \hat{a}_i$
Decompose $[U, S, V^T] = \mathsf{svd}(H)$
Build rotation $R = UV^T$
Rotate $\tilde{B} = \hat{B}R$ so each $\tilde{b}_i = \hat{b}_i R$
return $\hat{A}$, $\tilde{B}$

Scaling In some settings, it makes sense to align datasets by scaling one of them to fit better with the other, formulated as $(R^*, t^*, s^*) = \text{argmin}_{s\in\mathbb{R}, R\in\mathbb{SO}(d)} \sum_{i=1}^{n} \|a_i - s(b_i - t)R\|^2$. In addition to the choices of translation and rotation, the optimal choice of scaling can also be decoupled.

Horn et al. [18] introduced two mechanisms for solving for a scaling that minimizes RMSE. Assuming the optimal rotation R^* has already been applied to obtain $\hat{B}$, then a closed-form solution for scaling is $s^* = \sum_{i=1}^{n} \langle \hat{a}_i, \hat{b}_i \rangle / \|\hat{B}\|_F^2$. The sketch for absolute orientation with scaling, is in Algorithm 4.

Algorithm 4 AO+SCALING(A, B)

$\tilde{B} \leftarrow$ AO- ROTATION(A, B)
Compute scaling $s = \sum_{i=1}^{n} \langle a_i, \tilde{b}_i \rangle / \|\tilde{B}\|_F^2$
return $\breve{B}$ as $\breve{B} \leftarrow s\tilde{B}$ so for each $\breve{b}_i = s\tilde{b}_i$.

The steps of rotation, scaling, and translation fit together to give us Algorithm 5.

Algorithm 5 AO- CENTERED+SCALING(A, B)

Compute $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$ and $\bar{b} = \frac{1}{n}\sum_{i=1}^{n} b_i$
Center $\hat{A} \leftarrow (A, \bar{a})$ so each $\hat{a}_i = a_i - \bar{a}$, and similarly $\hat{B} \leftarrow (B, \bar{b})$
Compute the sum of outer products $H = \sum_{i=1}^{n} \hat{b}_i^T \hat{a}_i$
Decompose $[U, S, V^T] = \mathsf{svd}(H)$
Build rotation $R = UV^T$
Rotate $\tilde{B} = \hat{B}R$ so each $\tilde{b}_i = \hat{b}_i R$
Compute scaling $s = \sum_{i=1}^{n} \langle a_i, b_i \rangle / \|B\|_F^2$
Scale $\breve{B}$ as $\breve{B} \leftarrow s\tilde{B}$ so for each $\breve{b}_i = s\tilde{b}_i$.
return $\tilde{A}$, $\breve{B}$

Horn et al. [18] presented an alternative closed-form choice of scaling s which minimizes RMSE, but under a slightly different situation. In this alternate formulation, A must be scaled by $1/s$ and B by s, so the new scaling is somewhere in the (geometric) middle of that for A

and B. We found this formulation less intuitive, since the RMSE is dependent on the scale of the data, and in this setting the new scale is aligned with neither of the datasets. However, Horn et al. [18] only showed that the choice of optimal scaling is invariant from the rotation in the second (less intuitive) variant. We present a proof that this rotation invariance also holds for the first variant. The proof uses the structure of the SVD-based solution for optimal rotation, with which Horn et al. may not have been familiar.

Lemma 1 *Consider two points sets A and B in $\mathbb{R}^d$. After the rotation and scaling in Algorithm 4, no further rotation about the origin of $\check{B}$ can reduce the RMSE.*

Proof We analyze the SVD-based approach we use to solve for the new optimal rotation. Since we can change the order of multiplication operations of sb_iR, i.e., scale then rotate, we can consider first applying s^* to B, and then re-solving for the optimal rotation. Define $\check{B} = s^*B$, so each $\check{b}_i = s^*b_i$. Now to complete the proof, we show that the optimal rotation $\check{R}$ derived from A and $\check{B}$ is the same as was derived from A and B.

Computing the outer product sum $\check{H} = \sum_{i=1}^n \check{b}_i^T a_i = \sum_{i=1}^n (s^*b_i)^T a_i = s^* \sum_{i=1}^n b_i^T a_i = s^*H$, is just the old outer product sum H scaled by s^*. Then, its SVD is $\mathsf{svd}(\check{H}) \rightarrow [\check{U}, \check{S}, \check{V}^T] = [U, s^*S, V^T]$, since all of the scaling is factored into the S matrix. Then, since the two orthogonal matrices $\check{U} = U$ and $\check{V} = V$ are unchanged, we have that the resulting rotation $\check{R} = \check{U}\check{V}^T = UV^T = R$ is also unchanged. □

Preserving inner products While Euclidean distance is a natural measure to preserve under a set of transformations, many word vector embeddings are evaluated or accessed by Euclidean inner product operations. It is natural to ask if our transformations also maximize the sum of inner products of the aligned vectors. Or does it maximize the sum of cosine similarity: the sum of inner products of *normalized* vectors. Indeed we observe that AO- ROTATION(A, B) results in a rotation $\tilde{R} = \text{argmax}_{R \in \mathbb{SO}(d)} \sum_{i=1}^n \langle a_i, b_i R \rangle$.

Lemma 2 AO- ROTATION(A, B) *rotates B to $\tilde{B}$ to maximize $\sum_{i=1}^n \langle a_i, \tilde{b}_i \rangle$. If $a_i \in A$ and $b_i \in B$ are normalized $\|a_i\| = \|b_i\| = 1$, then the rotation maximizes the sum of cosine similarities $\sum_{i=1}^n \left\langle \frac{a_i}{\|a_i\|}, \frac{\tilde{b}_i}{\|\tilde{b}_i\|} \right\rangle$.*

Proof From Hanson and Norris [15] we know AO- ROTATION(B) finds a rotation R^* so

$$R^* = \underset{R \in \mathbb{SO}(d)}{\text{argmin}} \sum_{i=1}^n \|a_i - (b_i R)\|^2.$$

Expanding this equation, we find

$$R^* = \underset{R \in \mathbb{SO}(d)}{\text{argmin}} \left(\sum_{i=1}^n \|a_i\|^2 - \sum_{i=1}^n 2\langle a_i, b_i R \rangle + \sum_{i=1}^n \|b_i R\|^2 \right).$$

Now, the length of a vector does not change upon rotation(R), thus, $\|b_i R\|^2 = \|b_i\|^2$. So, since $\|a_i\|^2$ and $\|b_i\|^2$ are both lengths of vectors and thus, properties of the dataset, they do not depend on the choice of R and as desired

$$R^* = \text{argmax}_{R \in \mathbb{SO}(d)} \sum_{i=1}^n \langle a_i, b_i R \rangle.$$

If all a_i, b_i are normalized, then R does not change the norm $\|\tilde{b}_i\| = \|b_i R\| = \|b_i\| = 1$. So for $\tilde{b}_i = b_i R$, each $\langle a_i, \tilde{b}_i \rangle = \langle \frac{a_i}{\|a_i\|}, \frac{\tilde{b}_i}{\|\tilde{b}_i\|} \rangle$ and hence, as desired,

$$R^* = \text{argmax}_{R \in \mathbb{SO}(d)} \sum_{i=1}^{n} \left\langle \frac{a_i}{\|a_i\|}, \frac{b_i R}{\|b_i R\|} \right\rangle.$$

□

Several evaluations of word vector embeddings use cosine similarity, so it suggests first normalizing all vectors $a_i \in A$ and $b_i \in B$ before performing AO- ROTATION(A, B). However, we found this does not empirically work as well. The rational is that vectors with larger norm tend to have less noise and are supported by more data. So the unnormalized alignment effectively weights the importance of aligning the inner products of these vectors more in the sum, and this leads to a more stable method. Hence, in general, we do not recommend this normalization preprocessing.

2.1 Extension to contextualized embeddings

In recent years, contextualized embeddings such as ELMo [31] and BERT [8] have become increasingly popular, because of their ability to express the polysemity of words. A word in these frameworks is not expressed as a single vector, but rather, based on its different meanings or different contexts it has been used in. That is, each instance of a word is represented by a different vector in the embedding space. Our method to align individual vectors does not directly apply in this scenario.

We propose a simple extension to handle this scenario. Given a word w_i with two separate contextual embeddings, let these embedding vectors be two sets $A_i = \{a_{i,1}, a_{i,2}, \ldots a_{m_{A,i}}\}$ and $B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{m_{B,i}}\}$ of sizes $m_{A,i}$ and $m_{B,i}$, respectively. Then, our method, instead of aligning a single pair of vectors for each word, it aligns *all* vector pairs for each word. For instance, for finding the optimal rotation, this involves an alignment for n words, each ith word w_i, then the outer product matrix is defined

$$H = \sum_{i=1}^{n} \frac{1}{m_{A,i} m_{B,i}} \sum_{j=1}^{m_{A,i}} \sum_{j'=1}^{m_{B,i}} a_{i,j}^T b_{i,j'},$$

where each set of all-pairs is weighted equally for each i (this is accomplished by dividing by the number of such pairs $m_{A,i} m_{B,i}$.)

This all-pairs alignment can be computationally expensive as the number of instances of each word $m_{A,i}$ and $m_{B,i}$ increase; even if we only use 10 instances of each word, in each embedding, this increases the number of alignments by a factor 100. However, we observe, in each step the set A_i and B_i can be replaced by their averages

$$\bar{a}_i = \frac{1}{m_{A_i}} \sum_{j=1}^{m_{A,i}} a_{i,j} \text{ and } \bar{b}_i = \frac{1}{m_{B_i}} \sum_{j=1}^{m_{B,i}} b_{i,j}.$$

Then, the overall means $\bar{b}, \bar{a}$, outer product H, and scaling s are the same using all instances or the mean instance.

Lemma 3 *The alignments found using all-pair alignment when each word has multiple instances in each embedding is equivalent to that computed by aligning the averages of each set of instances.*

Proof We need to analyze the 4 quantities computed as part of any transformation: the two averages, the outer product, and the scale. In short, these are all linear vector operations (sum, outer product, inner product), so a vector average can be factored out.

For each average

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m_{A,i}}\sum_{j=1}^{m_{A,i}} a_{i,j} = \frac{1}{n}\sum_{i=1}^{n}\bar{a}_i,$$

and similarly for $\bar{b}$, the calculations are equivalent.

For the outer product

$$\begin{aligned} H &= \sum_{i=1}^{n}\frac{1}{m_{A,i}m_{B,i}}\sum_{j=1}^{m_{A,i}}\sum_{j'=1}^{m_{B,i}} a_{i,j}^T b_{i,j'} \\ &= \sum_{i=1}^{n}\left(\frac{1}{m_{A,i}}\sum_{j=1}^{m_{A,i}} a_{i,j}\right)^T\left(\frac{1}{m_{B,i}}\sum_{j'=1}^{m_{B,i}} b_{i,j'}\right) \\ &= \sum_{i=1}^{n}\bar{a}_i^T\bar{b}_i. \end{aligned}$$

And finally for the scale

$$\begin{aligned} s &= \sum_{i=1}^{n}\frac{1}{m_{A,i}m_{B,i}}\sum_{j=1}^{m_{A,i}}\sum_{j'=1}^{m_{B,i}}\langle a_{i,j}, b_{i,j'}\rangle / \|B\|_F^2 \\ &= \sum_{i=1}^{n}\left\langle\frac{1}{m_{A,i}}\sum_{j=1}^{m_{A,i}} a_{i,j}, \frac{1}{m_{B,i}}\sum_{j'=1}^{m_{B,i}} b_{i,j'}\right\rangle / \|B\|_F^2 \\ &= \sum_{i=1}^{n}\langle\bar{a}_i, \bar{b}_i\rangle / \|B\|_F^2, \end{aligned}$$

where

$$\|B\|_F^2 = \sum_{i=1}^{n}\left\|\frac{1}{m_{B,i}}\sum_{j=1}^{m_{B,i}} b_{i,j}\right\|^2 = \sum_{i=1}^{n}\|\bar{b}_i\|^2.$$

Note that the normalization term $\|B\|_F^2$ is defined on the average sum of instances for the all-pairs version, since this is a quadratic operation, and otherwise does not factor out. □

2.2 Related approaches

As mentioned, Smith et al. [40] use Algorithm 1 to align word2vec word embeddings on English and Italian corpuses, and show that this simple approach is effective in translation. Our work can be seen as building on this, in that we show how to interpret the intrinsic accuracy of such an alignment, how to align word vector corpuses created by different mechanisms, and when to use which variant of the closed-form solutions. Additionally, we confirm some of their language translation results and show that it extends to when the embedding mechanisms

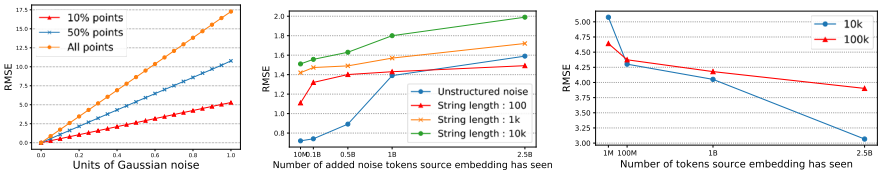

Fig. 1 RMSE Error after noise and AO- ROTATION alignment: Left: adding Gaussian noise to 10%, 50% or all points. Middle: adding structured and unstructured noise before embedding. Right: incrementally added data

for the different language corpuses are not the same (e.g., one by word2vec and one by GloVe), as demonstrated in Sect. 4.2.

There are several other methods in the literature which attempt to jointly compute embeddings of datasets so that they are aligned, for instance in jointly embedding corpuses in multiple languages [16,26]. The goal of the approaches we study is to circumvent these more complex joint alignments. A couple of very recent papers propose methods to align embeddings after their construction, but focus on *affine transformations*, as opposed to the more restrictive but distance preserving rotations of our method. Bollegala et al. [5] use gradient descent, for parameter γ, to directly optimize

$$\underset{M \in \mathbb{R}^{d \times d}}{\operatorname{argmin}} \sum_{i=1}^{n} \|a_i - b_i M\|^2 + \gamma \|M\|_F^2.$$

Another approach, by Sahin et al. [36], uses Low Rank Alignment (LRA), an extension of aligning manifolds from LLE [24]. This approach has a 2-step but closed-form solution to find an affine transformation applied to both embeddings simultaneously. Neither approach directly optimizes for the optimal transformation, and requires regularization parameters; this implies if embeddings start far apart, they remain further apart than if they start closer. Both find affine transformations M over $\mathbb{R}^{d \times d}$, not a rotation over the non-convex $\mathbb{O}(d)$ as does our approach. This changes the Euclidean distance found in the original embedding to a Mahalanobis distance that will change the order of nearest neighbors under Euclidian and cosine distance. Finally, the LRA approach requires an eigendecomposition of an $2n \times 2n$ matrix, where as ours only requires this of a $d \times d$ matrix, so LRA is far less scalable.

3 Evaluating accuracy of variants

We evaluate the effectiveness of our methods on a variety of scenarios, typically using the Root Mean Square Error: $\mathsf{RMSE}(A, B) = \sqrt{\frac{1}{|A|} \sum_{i=1}^{|A|} \|a_i - b_i\|^2}$. We fix the embedding dimension of each A (the target) and B (the source) at 300, and assume $|A| = |B| = n = 100{,}000$ or in some cases $n' = |A| = |B| = 10{,}000$.

We consider embeddings with GloVe [5] (our default), or word2vec [25,27] with Gensim [34], or occasionally RAW which is just the L^1 normalized word count vectors embedded with SVD [20]. We obtain all these three embeddings for our experiments using our default dataset is the 4.57 billion token English Wikipedia dump, which we found to be made up of $243K$ vocabulary words (distinct tokens). For word2vec, the Gensim [34] library provides code for obtaining embeddings of a desired dimensionality, and for GloVe, the code [5] is provided by the authors themselves. To obtain RAW embeddings, we run a simple bag of

words model which enumerates for each word, how many times it appeared with other words in the vocabulary in a sentence, to give us a vector representation for the word. The RAW word vectors, thus have the same dimension as that of the vocabulary itself. This when normalized captures the pointwise mutual information and is called the Pointwise Mutual Information (PMI) Matrix. After embedding using each of these three mechanisms, we select the top 100K most frequent words and their corresponding embeddings for our experiments.

We compare against existing GloVe embeddings of Wikipedia + Gigaword (G(WG), 6 billion tokens, 400K vocab), Common Crawl (G(CC42), 42 billion tokens, 1.9M vocab), and Common Crawl (G(CC840), 840 billion tokens, 2.2M vocab), and the existing word2vec embedding of Google News (W(GN), 100 billion tokens, 3 million vocab). All of these embeddings are available online (https://nlp.stanford.edu/projects/glove/, https://code.google.com/archive/p/word2vec/) and were downloaded.

When aligning GloVe embeddings to other GloVe embeddings we use AO- ROTATION. When aligning embeddings from different sources we use AO+SCALING.

Default data settings In each embedding, we always consider a consistent vocabulary of $n = 100{,}000$ words. To decide on this set, we found the n most frequent words used in the default Wikipedia dataset and that were embedded by GloVe. In one case, we compare against smaller datasets and then only work with a small vocabulary of size $n' = 10{,}000$ found the same way.

For each embedding, we represent each word as a vector of dimension $d = 300$. Note that RAW originally uses an n-dimensional vector. We reduce this to d-dimensions by taking its SVD, as advocated by Levy et al. [20]. They demonstrate how word2vec implicitly captures the information as the Shifted Pointwise Mutual Information Matrix (SPMI) in low dimensions. They further demonstrate that computing the SVD of the SPMI matrix maintains the structure captured by the full dimensional matrix.

3.1 Calibrating RMSE

In order to make sense of the meaning of an RMSE score, we calibrate it to the effect of some easier to understand distortions. To start, we make a copy of A (the default G(W) embedding—we use this notation to signify a GloVe embedding $\mathsf{G}(\cdot)$ or the default Wikipedia corpus W)) and apply an arbitrary rotation, translation, and scaling of it to obtain a new embedding B. Invoking $\hat{A}, \hat{B} \leftarrow$ AO- CENTERED+SCALING(A, B), we expect that $\mathsf{RMSE}(\hat{A}, \hat{B}) = 0$; we observe RMSE values on the order of 10^{-14}, indeed almost 0 withstanding numerical rounding.

Gaussian noise Next we add Gaussian noise directly to the embedding. That is we define an embedding B so that each $b_i = a_i + g_i$ where $g_i \sim N_d(0, \sigma I)$, where $N_d(\mu, \Sigma)$ is a d-dimensional Gaussian distribution, and σ is a standard deviation parameter. Then, we measure $\mathsf{RMSE}(\hat{A}, \hat{B})$ from $\hat{A}, \hat{B} \leftarrow$ AO- ROTATION(A, B). Figure 1 (left) shows the effects for various σ values, and also when only added to 10% and 50% of the points. We observe the noise is linear, and achieves an RMSE of 2 to 5 with $\sigma \in [0.1, 0.3]$.

Noise before embedding. Next, we append noisy, *unstructured* text into the Wikipedia dataset with 1 billion tokens. We specifically do this by generating random sequences of m tokens, drawn uniformly from the $n = 10$K most frequent words; we use $m = \{0.01, 0.1, 0.5, 1, 2.5\}$ billion. We then extract embeddings for the same vocabulary of $n = 100$K words as before, from both datasets, and use AO- ROTATION to linearly transform the noisy one to the one without noise. As observed in Fig. 1 (middle), this only changes

Table 1 RMSE after alignment for embeddings

→	G(W)	G(WG)	G(CC42)	G(CC840)
G(W)	–	4.56	5.167	6.148
G(WG)	4.56	–	5.986	6.876
G(CC42)	5.167	5.986	–	5.507
G(CC840)	6.148	6.876	5.507	–
→		RAW	GloVe	word2vec
RAW		–	4.12	14.73
GloVe		0.045	–	12.93
word2vec		0.043	3.68	–
Scale to GloVe		25	1	0.25
Scale from GloVe		0.011	1	3

Top: Created from different datasets. Bottom: Created by different embeddings; uses AO+SCALING mapping rows onto columns and changing scale

from about 0.7 to 1.6 RMSE. The embeddings seem rather resilient to this sort of noise, even when we add more tokens than the original data.

We perform a similar experiment of adding structured text; we repeat a sequence made of $s = \{100, 1000, 10{,}000\}$ tokens of medium frequency so the total added is again $m = \{10\text{M}, 100\text{M}, 500\text{M}, 1\text{B}, 2.5\text{B}\}$. Again in Fig. 1 (middle), perhaps surprisingly, this only increases the noise slightly, when compared to the unstructured setting. This can be explained since only a small percentage of the vocabulary is affected by this noise, and by comparing to the Gaussian noise, when only added to 10% of the data, it has about a third of the RMSE as when added to all data.

Incremental data As a model sees more data, it is able to make better predictions and calibrate itself more accurately. This comes at a higher cost of computation and time. If after a certain point, adding data does not really affect the model, it may be a good trade-off to use a smaller dataset to make an embedding almost equivalent to the one the larger dataset would produce.

We evaluate this relationship using the RMSE values when a GloVe embedding from a smaller dataset B is incrementally aligned to larger datasets A using AO- ROTATION. We do this by starting off with a dataset of the first 1 million tokens of Wikipedia (1M). We then add data sequentially to it, to create datasets of sizes of 100M, 1B, 2.5B, or 4.57B tokens. For each dataset, we create GloVe embeddings. Then, we align each dataset using AO- ROTATION(A, B) where A (the target) is always the larger of the two datasets, and B (the source) is rotated and is the smaller of the two.

Figure 1 (right) shows the result using a vocabulary of $n = 100\text{K}$ and $n' = 10\text{K}$. The small n' is also used since for smaller datasets, many of the top 100K words are not seen. We observe that even this change in dataset size, decreasing from 4.57B tokens to 2.5B still results in substantial RMSE. However aligning with fewer but better represented words starts to show better results, supporting use of weighted variants.

3.2 Changing datasets and embeddings

Now with a sense of how to calibrate the meaning of RMSE, we can investigate the effect of changing the dataset entirely or changing the embedding mechanism.

Table 2 Spearman coefficient scores for synonym and analogy tests between the aligned word2vec to GloVe embeddings and between GloVe embeddings of Wikipedia and CC42 dataset

Test sets	GloVe	word2vec	word2vec to GloVe							
			Untransformed	r	$r+s$	$r+t$	$r+s+t$	Normalized	$w(r)$	$w(r+s+t)$
RG	0.614	0.696	0.041	0.584	0.584	0.570	0.594	0.553	0.592	**0.597**
WSIM	0.623	0.659	0.064	0.624	0.624	0.611	0.652	0.604	0.657	**0.664**
MC	0.669	0.818	0.013	0.868	0.868	0.843	0.873	0.743	0.878	**0.882**
SIMLEX	0.296	0.342	0.012	0.278	0.278	0.269	0.314	0.261	0.314	**0.316**
SYN	0.587	0.582	0.000	0.501	0.525	0.517	0.528	0.493	0.535	**0.539**
SEM	0.691	0.722	0.0001	0.624	0.656	0.633	0.697	0.604	0.702	**0.712**
Test sets	G(W)	G(CC42)	G(W) to G(CC42)							
			Untransformed	r	$r+s$	$r+t$	$r+s+t$	Normalized	$w(r)$	$w(r+s+t)$
RG	0.614	0.817	0.363	0.818	0.818	0.811	0.821	0.815	0.818	**0.825**
WSIM	0.623	0.63	0.017	0.618	0.618	0.615	0.618	0.601	0.616	**0.637**
MC	0.669	0.786	0.259	0.766	0.766	0.732	0.768	0.705	0.771	**0.774**
SIMLEX	0.296	0.372	0.035	0.343	0.343	0.339	**0.346**	0.296	**0.346**	**0.346**
SYN	0.587	0.625	0.00	0.566	**0.576**	0.572	**0.576**	0.502	**0.576**	**0.576**
SEM	0.691	0.741	0.00	0.676	0.684	0.676	0.688	0.565	0.690	**0.695**

r, s, and t stand for the functions of optimal rotation, scaling, and translation, respectively, and $w()$ is the weighted version of that function. When computing these scores across two embeddings, the best values are printed in bold

Dependence of datasets Table 1 (top) shows the RMSE when the 4 GloVe embeddings are aligned with AO- ROTATION, either as a target or source. The alignment of G(W) and G(WG) has less error than either to G(CC42) and G(CC840), likely because they have substantial overlap in the source data (both draw from Wikipedia). In all cases, the error is roughly on the scale of adding Gaussian noise with $\sigma \in [0.25, 0.35]$ to the embeddings, or reducing the dataset to 10M to 100M tokens. This is much more alignment error than in other experiments, indicating that the change in the source dataset (and likely its size) has a much larger effect than the embedding mechanism.

Dependence on embedding mechanism We now fix the dataset (the default 4.57B Wikipedia dataset W), and observe the effect of changing the embedding mechanism: using GloVe, word2vec, and RAW. We now use AO+SCALING instead of AO- ROTATION, since the different mechanisms tend to align vectors at drastically different scales.

Table 1 (bottom) shows the RMSE error of the alignments; the columns show the target (A) and the rows show the source dataset (B). This difference in target and source is significant because the scale inherent in these alignments change, and with it, so does the RMSE. Also as shown, the scale parameter s^* from GloVe to word2vec in AO+SCALING is approximately 3 (and non-symmetrically about 0.25 in the other direction from word2vec to GloVe). This means for the same alignment, we expect the RMSE to be between 3 to 4 ($\approx 1/0.25$) times larger as well.

However, with each column, with the same target scale, we can compare alignment RMSE. We observe the differences are not too large, all roughly equivalent to Gaussian noise with $\sigma = 0.25$ or using only 1B to 2.5B tokens in the dataset. Interestingly, this is less error that changing the source dataset; consider the GloVe column for a fair comparison. This corroborates that the embeddings find some common structure, capturing the same linear structures, analogies, and similarities. And changing the datasets is a more significant effect.

3.3 Similarity and analogies after alignment

The GloVe and word2vec embeddings both perform well under different benchmark similarity and analogy tests. These results will be unaffected by rotations or scaling. Here we evaluate how these tests transfer under alignment. Using the default Wikipedia dataset, we use several variants of ABSOLUTEORIENTATION to align GloVe and word2vec embeddings. Then, given a synonym pair (i, j) we check whether $b_j \in B$ (after alignment) is in the neighborhood of a_i.

More specifically, we use 4 common similarity test sets, which we measure with cosine similarity [21]: Rubenstein–Goodenough (RG, 65 word pairs) [35], Miller–Charles (MC, 30 word pairs) [28], WordSimilarity-353 (WSIM, 353 word pairs) [12], and SimLex-999 (SIMLEX, 999 word pairs) [17]. We use the Spearman correlation coefficient (in $[-1, 1]$, larger is better) to aggregate scores on these tests; it compares the ranking of cosine similarity of a_i to the paired aligned word b_j, to the rankings from a human-generated similarity score.

Table 2 shows the scores on just the GloVe and word2vec embeddings, and then across these aligned datasets. To understand how the variants of ABSOLUTEORIENTATION compare, we compute the scores after each of the various optimal transformation types are applied: rotation, then scaling, then translation, and finally we consider if we normalize all vectors before alignment to maximize cosine similarities. Before transformation ("untransformed") the across-dataset comparison is very poor, close to 0; that is, extrinsically there is very little information carried over. However, alignment with just AO- ROTATION achieves scores nearly as good as, and sometimes better than on the original datasets. word2vec scores higher than GloVe, and the across-dataset scores are typically between these two scores. Adding scaling with AO+SCALING has no effect on the scores on the similarity test because they

Table 3 Synonym and Analogy scores from rotations (applied to all words) learned on 100K, top 10K words, and top 10K words normalized

	AO+R	AO+R	AO- NORMALIZED
	100K	10K	10K
RG	0.584	0.576	0.588
WSIM	0.624	0.612	0.643
MC	0.868	0.817	0.851
SIMLEX	0.278	0.292	0.308
SYN	0.501	0.505	0.511
SEM	0.624	0.616	0.616

are measured with cosine similarity. However, also applying the optimal translation does increase the scores even though it optimizes Euclidean distance and not cosine distance. Perhaps surprisingly, applying rotation along with translation *and* scaling improves more than just applying rotation and translation. This method applies scaling after the dataset is centered, so this then alters the inner products, and in a useful way.

We perform the same experiments on 2 Google analogy datasets [27]: SEM has 8869 analogies and SYN has 10675 analogies. These are of the form "A:B::C:D" (e.g., "man:woman::king:queen"), and we evaluate across datasets by measuring if vector v_D is among the nearest neighbors in dataset A of vector $v_C + (v_B - v_A)$ in dataset B. The results are similar to the synonym tests, where AO- ROTATION alignment across-datasets performs similar to within either embedding, and scaling and rotation provided small further improvement. In this case, performing rotation and scaling improves upon just rotation. This is because the analogies are accessing something more complicated about the embedding, and so adjusting the scale more aligns the Euclidean distance and hence the vector structure needed to succeed in analogies.

The right part of the table shows the effect of various weightings. Normalization makes the similarity and analogy scores worse, but weighting by the norms consistently increases the scores. Moreover, also scaling and rotating (e.g., as w(r+s+t)) improves the scores further.

We also align G(W) to G(CC42), to observe the effect of only changing the dataset. The G(CC42) dataset performs better itself; it uses more data. The small similarity tests (RG,MC) show some extrinsic information is captured without any alignment, but otherwise across-embedding scores have a similar pattern to across-dataset scores.

Next, in Table 3, we further investigate the effect of various weightings (or normalizing) before alignment. In these tests, we show the effect on AO- ROTATION with three types of weighting. As before we simply apply AO- ROTATION on all 100K words. But we also find the optimal R on only the most frequent 10K words using AO- ROTATION, and then again using AO- NORMALIZED on just these 10K words. The rotation and evaluation are still on all 100K words needed for the tests. Surprisingly AO- NORMALIZED(10K) performs better than AO- ROTATION(10K), and comparably to AO- ROTATION(100K). This indicates that similarity optimization is useful when the words all have sufficient data to embed them properly.

3.4 Comparison to baselines

Next, we perform similarity tests to compare against alignment implementations of methods by Sahin et al. [36] (LRA) and Bollegela et al. [5] (Affine Transformations). We reimple-

Table 4 Modified similarity tests (on only top 10K words) after alignment by Affine Transformation (AffTrans), LRA, and AO- ROTATION of Wikipedia and CC42 GloVe embeddings

	LRA	AffTrans	AO+R	AO+R
Test sets	10K	10K	10K	100K
RG	0.701	0.301	0.728	0.818
WSIM	0.616	0.269	0.612	0.618
MC	0.719	0.412	0.722	0.766
SIMLEX	0.327	0.126	0.340	0.343

Table 5 RMSE variation with word frequency in (a) GloVe Wiki to GloVe Common Crawl and (b) word2vec to GloVe evaluated for Wiki dataset

Word	Frequency in Wiki	Norm in (GloVe) Wiki	Wiki to CC (42B)	word2vec to GloVe
talk	187513532	9.681	8.608	12.462
november	2340726	7.847	5.26	8.614
man	1035008	8.648	4.25	5.161
statistical	83531	5.891	4.63	5.097
bubbles	11200	5.455	4.66	3.768
skateboard	3804	5.670	5.714	3.891
emoji	1761	6.090	6.781	2.402
haymaker	705	4.108	5.951	1.573

Table 6 Similarity and analogy tests before and after alignment and combining embeddings derived from different techniques and datasets by AO+CENTERED+WEIGHTED

Test sets	G(W)	W(W)	[G(W)⊙W(W)]	W(GN)	[G(W)⊙W(GN)]	G(CC840)	[G(CC840)⊙W(GN)]
RG	0.614	0.696	0.716	0.760	**0.836**	0.768	0.810
WSIM	0.623	0.659	0.695	0.678	0.708	0.722	**0.740**
MC	0.669	0.818	**0.869**	0.800	0.811	0.798	0.847
SIMLEX	0.296	0.342	0.368	0.367	0.394	0.408	**0.446**
SYN	0.587	0.582	0.592	0.595	0.607	**0.618**	0.609
SEM	0.691	0.722	**0.759**	0.713	0.733	0.729	0.733

Best scores in **bold**

mented their algorithms, but did not spend significant time to optimize the parameters; recall our method requires no hyperparameters. We only used the top $n' = 10$K words for these transformations because these other methods were much more time and memory intensive. We only computed similarities among pairs in the top 10K words for fairness (about two-thirds of the word pairs evaluated, so the scores do not match other tables), and did not perform analogy tests since fewer than one-third of analogies fully showed up in the top 10K. Table 4 shows results for aligning the G(W) and G(CC42) embeddings with these approaches. Our ABSOLUTEORIENTATION-based approach does significantly better than Bollegela et al.'s [5] Affine Transformations and generally better than Sahin et al.'s [36] LRA. Our advantage over LRA increases when aligning all $n = 100$K words; by comparison, LRA ran out of memory since it requires an $n \times n$ dense matrix decomposition.

3.5 Dependence of RMSE variation with word frequency

Table 5 shows some sampled words of various frequencies in the Wikipedia dataset. A word that is more frequently seen in a corpus is generally seen with a larger proportion of other words and contexts, and thus as observed in the table, has a vector representation that has larger norm than a word which has low frequency. This results in the contribution of high frequency words in the rotation matrix H, computed for minimizing the RMSE, to also be larger. This larger frequency, and larger norm, also manifests itself in the error after alignment, as shown in the last two columns of Table 5, both between data sets and between embedding mechanisms. The relation in the amount of RMSE between words appears even more correlated when between embedding mechanisms (in this case word2vec and GloVe). The low-frequency words likely exhibit some baseline noise in the case with different datasets (Wiki and CC(42B)), which obscures this relationship for low-frequency words.

3.6 Discussion on the right variant

Most of the gain using AbsoluteOrientation is achieved by just finding the optimal rotation R with AO- Rotation. However, consistent improvement can be found by weighting the large points more using AO+Weighted and by applying translation or scaling, and slightly more by applying both.

When different datasets are aligned using the same mechanism (e.g., both with GloVe or both with word2vec), then it is debatable whether scaling and translation is necessary, since scaling does not affect cosine similarity, and translation changes intrinsic inner product properties. However, using a weighting to put more weight on longer (and implied more robustly embedded) words does not alter any intrinsic properties, and only seems to create better alignments.

When datasets are embedded with different mechanisms (e.g., one with word2vec and one with GloVe) then they are not scaled properly with respect to each other. In this case, it is important to find the optimal scaling to put them in a consistent interpretable scale, and to ensure analogy relations are optimized. So we strongly recommend using scaling in this setting.

4 Embedding alignment : applications

We highlight a few applications which may be served by this alignment, and comparison mechanisms that we design and demonstrate their effectiveness.

4.1 Boosting via ensembles

A direct application of combining different embeddings can be to increase its robustness. We show that ensembles of pre-computed word embeddings found via different mechanisms and on different datasets can boost the performance on the similarity and analogy tests beyond that of any single mechanism or dataset. The boosting strategy we use here is just simple averaging of the corresponding words after the embeddings have been aligned.

Table 6 shows the performance of these combined embedding in three experiments. The first set shows the default Wikipedia dataset under GloVe (G(W)), under word2vec (W(W)), and combined ([G(W)⊙W(W)]). The second set shows word2vec embedding of GoogleNews (W(GN)),

Table 7 The 5 closest neighbors of a word before and after alignment by AO- ROTATION (between English–Spanish)

Word	Neighbors before alignment	Neighbors after alignment
woman	her, young, man, girl, mother	her, girl, **mujer**, mother, man
week	month, day, year, monday, time	days, **semana**, year, day, month
casa	apartamento, casas, palacio, residencia, habitaci	casas, **home**, homes, habitaci, apartamento
caballo	caballos, caballer, jinete, jinetes, equitaci	**horse**, horses, caballos, jinete
sol	sombra, luna, solar, amanecer, ciello	**sun**, moon, luna, solar, sombra

Target word (translation) in **bold**

Table 8 The 5 closest neighbors of a word before and after alignment by ABSOLUTEORIENTATION (between English–French)

Word	Neighbors before alignment	Neighbors after alignment
woman	her, young, man, girl, mother	her, young, man, **femme**, la
week	month, day, year, monday, time	month, day, year, **semaine**, start
heureux	amoureux, plaisir, rire, gens, vivre	**happy**, plaisir, loving, amoureux, rire
cheval	chein, petit, bateau, pied, jeu	**horse**, dog, chien, red, petit
daughter	father, mother, son, her, husband	mother, **fille**, husband, mere, her

Target word (translation) in **bold**

and combined ([G(W)⊙W(GN)]) with G(W). The third set shows GloVe embedding of Common-Crawl (840B) (G(CC840)) and then combined with W(GN) as [G(CC840)⊙W(GN)]. Combining two embeddings using AO+CENTERED+WEIGHTED consistently boosts the performance on similarity and analogy tests. Very similar boosting results occur independent of the precise alignment mechanism (e.g., using AO- CENTERED+SCALING). The best score on each experiment is in bold, and in 5 out of 6 cases, it is from a combined embedding. Moreover, except for this one case, the combined embedding always performs better on all tests that both of the individual embeddings, and in this one case, G(CC804)⊙W(GN) still outperforms W(GN) on SEM analogies. For instance, remarkably, G(W)⊙W(W) which only uses the default $4.57B$ token Wikipedia dataset, performs better or nearly as well as W(GN) which uses $100B$ tokens. Moreover, in some cases the improvement is significant; on the large similarities test SIMLEX, the [G(CC840)⊙W(GN)] score is 0.443 or 0.446 with weights, whereas the best score without boosting is only 0.408 using G(CC840).

4.2 Aligning embeddings across languages and embeddings

Word embeddings have been used to place word vectors from multiple languages in the same space [16,26]. These either do not perform that well in monolingual semantic tasks as noted in Luong, Pham and Manning [23] or use learned affine transformations [26], which distort distances and do not have closed-form solutions. Smith et al. [40] use the equivalent of AO- ROTATION to translate between word embeddings from different languages that have been extracted using the same method. We extend that here to verify that no matter the embedding mechanism, we can translate using a variant of ABSOLUTEORIENTATION. We use the ability to choose the right variant of absolute orientation as per Sect. 3.6 to orient different embeddings onto each other coherently. We use the default English GloVe embedding

from Wikipedia and the FastText https://github.com/facebookresearch/fastText embedding for Spanish. FastText is yet another unsupervised learning paradigm for obtaining vector representations for words which uses a lot of concepts from word2vec, skipgram models, and bag of words. As presented, these two have been derived using different methods and are thus oriented differently in 300-dimensional space. We extract the embeddings for the most frequent 5000 words from the default English Wikipedia dataset (that have translations in Spanish) and their translations in Spanish and align them using AO+CENTERED+WEIGHTED. We test before and after alignment, for each of these 10,000 words, if their translation is among their nearest 1, 5, and 10 neighbors. Before alignment, the fraction of words with its translation among its closest 1, 5, and 10 nearest neighbors is 0.00, 0.160, and 0.160, respectively, while after alignment it is 0.372, 0.623, and 0.726, respectively. Some examples of translations are in Table 7.

We perform a cross-validation experiment to see how this alignment applies to new words not explicitly aligned. On learning the rotation matrix above, we apply it to a set of 1000 new "test" Spanish words (the translations of the next 1000 most frequent English words) and bring it into the same space as that of English words as before. We test these 2000 new words in the embedded and aligned space of 12,000 words (now 6000 from each language). Before alignment, the fraction of times their translations are among the closest 1, 5, and 10 neighbors are 0.00, 0.00, and 0.00, respectively. After alignment it is 0.311, 0.689, and 0.747, respectively (comparable to results and setup in Mikolov et al. [26], using jointly learned affine transformations).

We perform a similar experiment between English and French, and see similar results. We first obtain 300-dimensional embeddings for English Wikipedia dump using GloVe, and for French words from the FastText embeddings. Then, we extract the embeddings for the most frequent 10,000 words from the default Wikipedia dataset (that have translations in French) and their translations in French and align them using AO+CENTERED+WEIGHTED. We test before and after alignment, for each of these 10,000 words, if their translation is among their nearest 1, 5, and 10 neighbors. Before alignment, the fraction of words with its translation among its closest 1, 5 and 10 nearest neighbors is 0.00, 0.054, and 0.054, respectively, while after alignment it is 0.478, 0.755, and 0.810, respectively. Table 8 lists some examples before and after translation.

We again perform a cross-validation experiment to see how this alignment applies to new words not explicitly aligned. On learning the rotation matrix above, we apply it to a set of 1000 new "test" French words (the translations of the next 1000 most frequent English words in the default dataset) and bring it into the same space as that of English words as before. We test in this space of 22,000 words now, if their translations are among the closest 1, 5 and 10 nearest neighbors of the 2000 new words (1000 French and their translations in English). Before alignment, the fraction of times their translations are among the closest 1, 5 and 10 neighbors are 0.00, 0.00, and 0.00, respectively. After alignment it is 0.307, 0.513, and 0.698, respectively.

4.3 Aligning multiple languages onto same space

As demonstrated in Sect. 4.2, pairwise alignment of words from different languages needs relatively few points to find the alignment to achieve good accuracy in translation between the two languages for a much larger set of words. This allows us to have a low cost operation to map words of one language to their corresponding translated words in the another language. This additionally leads us to a follow-up application. For many language pairs (say languages

Table 9 Translating Spanish to French by aligning directly as compared to aligning both to English

Top n accuracy	Unaligned	Pairwise aligned	Indirectly aligned
$n = 1$	0.0	0.312	0.277
$n = 5$	0.0	0.635	0.601
$n = 10$	0.0	0.723	0.707

L_1 and L_2), we might not have a known dictionary of corresponding word-pairs. In such cases, finding an alignment for enabling translation can be impeded. However, for each of these languages L_1 and L_2, if corresponding words to a third language L_3 is known, aligning both L_1 and L_2 onto L_3 also brings L_1 and L_2 into the same space. Thus, translation of words from L_1 and L_2 is enabled without having a set of corresponding seed words in them by which to define the alignment. Aligning multiple languages onto the same space can thus, aid in multi-way translation. Further, for low-resource languages or pairs of languages for whom, only a very small set of translations, i.e., few corresponding points are known, aligning each of these languages to a more common language with which a larger correspondence is known, can help translation.

To demonstrate this, we pick languages L_1 and L_2 to be Spanish and French, respectively. We also pick the common language L_3 to be English, to whose word embedding space we align L_1 and L_2 to. In Table 9, in the first column, we have Spanish to French translations before alignment. As expected, the top 1, 5 and 10 neighboring word accuracies (as evaluated in Sect. 4.2) are poor (in fact 0 accuracy). In the second column, we have accuracies after aligning them onto each other using a pool of 2000 words for which we know translations, i.e., their one-to-one correspondences. Next, in the third column, we align both Spanish and French onto English, using the same set of 2000 words and then compare the accuracies for translations from Spanish to French. We find that the top 1, 5, and 10 accuracies are comparable between columns 2 and 3. Thus Spanish–French translation was enabled by knowing Spanish–English and French–English associations. This multi-way translation enabled by a third language's association leads us to many possibilities of aligning low-resource languages to each other easily.

5 Discussion

We have provided simple, closed-form method to align word embeddings. Code can be found on github (https://github.com/sunipa/Abs-Orientation). It allows for transformations for any subset of translation, rotation, and scaling. These operations all preserve the intrinsic Euclidean structure which has been shown to give rise to linear structures which allows for learning tasks like analogies, synonyms, and classification. All of these operations also preserve the Euclidean distances, so it does not affect the tasks which are measured using this distance; note the scaling also scales this distance, but does not change its order. Our experiments indicate that the rotation is essential for a good alignment, and the scaling is needed to compare embeddings generated by different mechanisms (e.g., GloVe and word2vec) and while helpful, not necessarily when the dataset is changed. Also, the translation provides minor but consistent improvement.

We also show how to explicitly optimize cosine similarity by first normalizing all words—however, this does not perform as well as instead optimizing Euclidean distance. Rather we

propose to weight words in the alignment by their norms, and this further improves the alignment because it emphasizes the words which have more stable embeddings.

This alignment enables new ways that word embeddings can be compared. This has the potential to shed light on the differences and similarity between them. For instance, as observed in other ways, common linear substructures are present in both GloVe and word2vec, and these structures can be aligned and recovered, further indicating that it is a well-supported feature inherent to the underlying language (and dataset). We also show that changing the embedding mechanism has less of an effect than changing the dataset, as long as that dataset is meaningful. Unstructured noise added to the input dataset appears not to have much effect, but changing from the 4.57B token Wikipedia dataset to the 840B token Common Crawl dataset has a large effect.

Additionally, we show that by aligning various embeddings, their characteristics as measured by standard analogy and synonym tests can be transferred from one embedding to another. We also demonstrate that cross-language alignment can aid in word translation even when coming from completely different embedding mechanisms, even in a cross-validation setting. This cross-embedding mechanism alignment opens the door for many other types of alignment word embeddings with embeddings generated from graphs, images, or any other dataset which has some useful word labels.

Finally, we showed that we can "boost" embeddings without revisiting the (sometimes quite enormous) raw data. This is surprisingly effective in improving scores on similarity and analogy test, results in the best known scores from embeddings on these tests. For instance, on the SIMLEX analogy test we improve upon the best known scores by almost 10% in the Spearman correlation coefficient. There are many other potential applications of these techniques for aligning high-dimensional data embeddings. We propose some scenarios where they may be used in the following section.

5.1 Other applications

Here, we enumerate a few applications—we do not experiment on many of these due to the extreme computational cost of performing an analysis of the effect (i.e., the baseline approaches of not using our techniques can be prohibitively expensive, or too qualitative to effectively evaluate).

(1) Common Crawl is one of the largest textual data sources available. Moreover, it consistently gets updated to include the ever increasing data on the internet. Each of these datasets has over 800B tokens, and extracting embeddings from these can be computationally expensive. However, extracting embeddings from the additional data not included in the previous update of Common Crawl should be significantly less expensive. Aligning an embedding from just the new data, and performing a weighted average with the older larger one may work as well or better than the embedding made from scratch.
(2) A similar weighted average alignment can help with specialized data. Consider data from scientific journals only, or of domain-specific biomedical terms. Embeddings from just these datasets would be very specialized and each word would have a specific word sense based on the domain. Aligning these to a gigantic corpus can enrich the specialized domain-related regions on the larger embedding.
(3) Tags and phrases in English can be single words or a string of words. Orienting an embedding of tags/phrases along say Common Crawl using an intersection of the single words in the two datasets can help place multi-worded tags or phrases around words

related to them. This can help derive meaning from random or unknown phrases. Images also often are annotated with a set of tag words. So orienting a set of tags can help orient images meaningfully among words.

(4) These methods can also be applied to even more heterogeneous embeddings than discussed above. We can orient heterogeneous embeddings derived from a variety of methods, e.g., for graphs including node2vec [14] or DeepWalk [30], and others [6,9,13], images [2,22], and for kernel methods [32,33]. For instance, RDF data can contain shorthand query phrases like "president children spouse" which answers the question "who are the spouses and children of presidents?" By orienting each word along word embeddings from Common Crawl, this may help answer similar questions even more abstractly. Heterogeneous networks have a mixture of node types. If there is an intersection of some nodes (and node types) between any two embeddings (heterogeneous or homogeneous), we can orient them meaningfully.

(5) Customer data collected at a company over different years and subsidiaries can be embedded using different features (such as income bracket, credit score, and location depending on the company). Using common customers over the year, diverse sources and new users can be added meaningfully to the embedding and inferred about, without embedding all of the data points from scratch. Moreover, embedding the same users from different years and aligning them can also help deduce the change in their features over time.

Acknowledgements Thanks to NSF CCF-1350888, ACI-1443046, CNS-1514520, CNS-1564287, IIS-1816149, and NVidia Corporation.

References

1. Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of two 3-d points sets. IEEE Trans Pattern Anal Mach Intell 9(5):698–700
2. Bay H, Tuytelaars T, Van Gool L (2006) SURF: speeded up robust features. In: ECCV
3. Besl PJ, McKay ND (1992) A method for registration of 3-d shapes. IEEE Trans Pattern Anal Mach Intell 14(2):239–256
4. Bojanowski P, Grave E, Joulin A, Mikolov T (2016) Enriching word vectors with subword information
5. Bollegala D, Hayashi K, Kawarabayashi KI (2017) Learning linear transformations between counting-based and prediction-based word embeddings. PLoS ONE 12:e0184544
6. Cai H, Zheng VW, Chang KC (2017) A comprehensive survey of graph embedding: problems, techniques and applications. Technical report, arXiv:1709.07604
7. Chen Y, Medioni G (1992) Object modelling by registration of multiple range images. Image Vis Comput 10:145–155
8. Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. Proc NAACL-HLT 2019:4171–4186
9. Dong Y, Chawla NV, Swami A (2017) metapath2vec: scalable representation learning for heterogeneous networks. In: KDD
10. Eggert DW, Lorusso A, Fisher RB (1997) Estimating 3-d rigid body transformations: a comparison of four major algorithms. Mach Vis Appl 9:272–290
11. Faugeras OD, Hebert M (1983) A 3-d recognition and positioning algorithm using geometric matching between primitive surfaces. Proc Int Jt Conf Artif Intell 8:996–1002
12. Finkelstein L, Gabrilovich E, Matias Y, Rivlin E, Solan Z, Wolfman G et al (2002) Placing search in context: the concept revisited. ACM Trans Inf Syst 20:116–131
13. Goyal P, Ferrara E (2017) Graph embedding techniques, applications, and performance: a survey. Technical report, arXiV:1705.02801
14. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: KDD
15. Hanson RJ, Norris MJ (1981) Analysis of measurements based on the singular value decomposition. SIAM J Sci Stat Comput 27(3):363–373

16. Hermann KM, Blunsom P (2013) Multilingual distributed representations without word alignment. ArXiv e-prints
17. Hill F, Reichart R, Korhonen A (2015) Simlex-999: evaluating semantic models with (genuine) similarity estimation. Comput Linguist 41:665–695
18. Horn BKP (1987) Closed-form solution of absolute orientation using unit quaternions. J Opt Soc Am A 4:629–642
19. Joulin A, Grave E, Bojanowski P, Mikolov T (2016) Bag of tricks for efficient text classification
20. Omer L, Yoav G (2013) Neural word embedding as implicit matrix factorization. In: NIPS
21. Omer L, Yoav G (2014) Linguistic regularities of sparse and explicit word representations. In: CoNLL
22. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60:91–110
23. Luong MT, Pham H, Manning CD (2015) Bilingual word representations with monolingual quality in mind. In: NAACL-HLT, pp 151–159
24. Mahadevan S, Boucher T, Carey CJ, Dyar MD (2014) Aligning mixed manifolds. In: AAAI
25. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. Technical report, arXiv:1301.3781
26. Mikolov T, Le QV, Sutskever I (2013) Exploiting similarities among languages for machine translation. ArXiv e-prints
27. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: NIPS, pp 3111–3119
28. Miller G, Charles W (1998) Contextual correlates of semantic similarity. Lang Cogn Process 6:1–28
29. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: EMNLP
30. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: KDD
31. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. In: Proceedings of NAACL
32. Rahimi A, Recht B (2007) Random features for large-scale kernel machines. In: NIPS
33. Rahimi A, Recht B (2008) Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In: NIPS
34. Radim Ř, Petr S (2010) software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks, Valletta, Malta, May 2010, pp 45–50. ELRA. http://is.muni.cz/publication/884893/en
35. Rubenstein H, Goodenough JB (1965) Contextual correlates of synonymy. Commun ACM 8:627–633
36. Sahin CS, Caceres RS, Oselio B, Campbell WM (2017) Consistent alignment of word embedding models. ArXiv e-prints
37. Schönemann PH (1966) A generalized solution to the orthogonal procrustes problem. Psychometrika 31(1):1–10
38. Schwartz JT, Sharir M (1987) Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. Int J Robot Res 6(2):29–44 **(Summer 1987)**
39. Shi Q, Petterson J, Dror G, Langford J, Smola A, Vishwanathan SVN (2009) Hash kernels for structured data. JMLR 10:2615–2637
40. Smith SL, Turban DH, Hamblin S, Hammerla NY (2017) Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In: ICLR
41. Walker MW, Shao L, Volz RA (1991) Estimating 3-D location parameters using dual number quaternions. CVGIP: Image Underst 54(3):358–367
42. Weinberger K, Dasgupta A, Langford J, Smola A, Attenberg J (2009) Feature hashing for large scale multitask learning. In: ICML

Sunipa Dev is a Computing Innovation Fellow at UCLA. She completed her Ph.D. at the School of Computing at the University of Utah. Her research focuses on understanding the structure of language representations and leveraging that to isolate and decouple associations and concept subspaces within. Her thesis developed debiasing algorithms, metrics, and probes, which build a more holistic understanding of the impact that invalid and biased associations can have on real world tasks.

Safia Hassan completed her undergraduate degree at the University of Utah in Applied Mathematics and Computer Science. Her undergraduate thesis focused on exploring vector spaces of predictive word embeddings through an absolute orientation technique of their respective coordinates in high dimensions. She continued to pursue her graduate degree in Computer Science and now is working as a Software Engineer.

Jeff M. Phillips is an Associate Professor at the School of Computing and Director of the Utah Center for Data Science, at the University of Utah. He has been funded by an NSF GRF, CI Fellowship, and CAREER Award, and he works in machine learning, data mining, algorithms and geometry, and databases. He has also written an undergraduate textbook (mathfordata.github.io) focusing on the mathematical and geometric aspects of data analysis.