Summer Research Project Report

Danyi Liu
supervised by Dr. Haim Dubossarsky

September, 2019

1 Measuring Align-ability Between Embedding Spaces

1.1 Motivation

When inducing cross-lingual word embedding spaces through orthogonal projection, we observed that the efficacy of alignment, measured by the precision of word translation retrieval, varied significantly across different language pairs. Coming from a linguistic perspective, his observation motivated us to investigate whether the align-ability of embedding spaces of different languages can be described or measured using linguistic features.

1.2 Alignment Results of Various Language pairs

The first step of our experiment was to set up a pipeline for aligning word embedding spaces and evaluating bilingual lexicon induction (BLI) results. Alignment was achieved by performing Procrustes analysis to obtain an orthogonal matrix to project the source embeddings to the target space. As for evaluation, we created functions for computing both P@k (precision at rank k) and MAP (mean average precision, also known as mean reciprocal rank). Our experiments were performed with *fastText* pretrained embeddings [1] and ground truth bilingual dictionaries used in *MUSE* [2] and were able to replicate the results published in *MUSE* in all language pairs apart from *zh-en* and *en-zh*.

		en-es	es-en	en-fr	fr-en	en-de	de-en	en-ru	ru-en	en-zh	zh-en
MUSE	p@1	77.4	77.3	74.9	76.1	68.4	67.7	47.0	58.2	40.6	30.2
Replicated	p@1	76.9	77.3	74.5	75.7	67.3	67.1	46.1	58.2	**27.3**	**9.3**
	p@5	88.4	90.2	87.1	87.7	84.7	81.4	69.9	74.8	**49.7**	**21.0**

Table 1: word translation retrieval precision@k results for five language pairs

1.3 Polysemy

We hypothesized that one of the reasons alignment failed was due to polysemy, when a word form has more than one related senses. Different languages have different distributions of polysemy. As a result, the embedding vectors in language L_a do not have an one-to-one correspondence with those in L_b, posing a challenge in aligning the two embedding spaces through orthogonal projection. Procrustes analysis provides a closed-form matrix that minimises the Euclidean distance between the projected source embedding $X^{'}_{L_a}$ and the target embedding space X_{L_b}. Consider an example of minimising the average Euclidean (L_2) distance between an embedding vector $s_1^{L_a}$ in source space X_{L_a} and its multiple translations $t_1^{L_b}, t_2^{L_b}, t_3^{L_b}$ in target embedding space X_{L_b}, the projection of $s_1^{L_a}$, denoted as $p_1^{L_a}$, would be expected to end up somewhere in the middle between $t_1^{L_b}, t_2^{L_b}, t_3^{L_b}$ where it is possible that the nearest neighbor of $p_1^{L_a}$ is not any of $t_1^{L_b}, t_2^{L_b}, t_3^{L_b}$.

To test the effect of polysemy on aligning embedding spaces, we artificially induced polysemy to a language by shotlisting a list of central words and merging multiple words within a certain cosine similarity range to each central word in a monolingual corpus. For instance, in a case where *chair* is the central word and the cosine similarity between *chair* and either of *sofa* or *couch* lies within a given range, all the occurrences of *sofa* and *couch* would be substituted with *chair*. The embedding space trained on this synthetic corpus would then only include an embedding vector for *chair* (but not for *sofa* or *couch*) representing all the senses originally represented by these three words.
When deciding which words to merge, two factors were considered: cosine similarity between the words being merged together $\boldsymbol{c}$ and the number of words being merged $\boldsymbol{n}$ (which we call degree of polysemy). In creating one set of synthetic embeddings with induced polysemy, we only considered words within the frequency range of 300 to 10000, merging n words that have a cosine similiarity of c with a central word to the central word. Lacking information about how many common polysemous words there are in different languages, we created 250 synthetic polysemous words in all our experiments. In the resulted corpus, $250 * n$ words were replaced with 250 central words.
When training the embeddings, we used the set of hyperparameters that yielded the best monolingual word similarity when applied to the original unadapted corpus.

1.3.1 Tolerence of Embedding Spaces to Polysemy

Experimental Setup
In order to isolate the effect of polysemy while holding all other factors fixed, we attempted to align two synthetic embeddings, X_a and X_b, of the same language with differently induced polysemy. When comparing these two embedding spaces, we would observe both one-to-many and many-to-one polysemy.

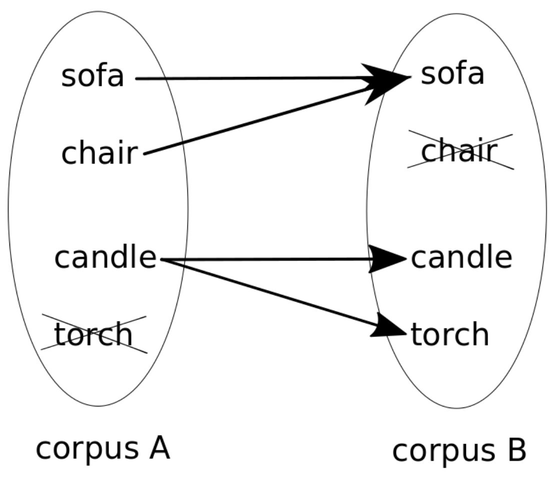


Figure 1: one-to-many and many-to-one polysemy

To align the two embedding spaces, we tried using (1) a dictionary made up of 5000 identical words that have not been merged in either corpus A or B (*book* - *book*) (2) a dictionary with half of the entries being words involved in the merging process (e.g. *sofa* - *sofa, chair* - *sofa*), the other half being identical words that were not involved in merging.
As for evaluation, five separate dictionaries were considered. For each of the dictionaries, an example of dictionary entry was given for the corpora pair shown in Figure 1.
(1) Self-alignment dictionary 1: *candle* - *candle*
(2) One-to-many polysemy dictionary: *candle* - *torch*
(3) Combined dictionary: *candle* - *[torch, candle]*
(4) Self-alignment dictionary 2: *sofa* - *sofa*

(5) Many-to-one polysemy dictionary: *chair* - *sofa*

Results

(i) Similarity between words being merged
We experimented with English, Spanish and Mandarin in this and the following section. We first fixed degree of polysemy to 2, i.e. merging two words to one, and created seven pairs of corpora A and B in which c ranged from 0.2 to 0.8. It turned out that the precision@1 for the combined dictionary remained perfect despite we were merging less and less similar words, i.e. the senses of the polysemy that we induced were more and more distant. The results remained the same regardless of whether the training dictionary we used contained words that were involved in merging. Although we did not observe any difference after including in merged words in the dictionary in our case, we did not formally evaluate the effect of using polysemy words for alignment, which could be worth exploring.

Another observation was that, when $n = 1$, the precision@1 values for self-alignment dictionary 1 and one-to-many polysemy, despite always summing up to one as we would expect since the precision@1 for the combined dictionary should be the sum of these two, were often quite similar without one constantly higher than the other across different c. Intuitively, which precision@1 was higher would depend on the relative frequency of the words being merged and the words they were merged to. It might be interesting to try to create a model that predicts the embedding vector resulted from embedding vectors whose corresponding words are merged together by taking their original embedding vectors and occurrence frequencies as input variables.

(ii) Degree of Polysemy
When we increased the degree of polysemy, merging 2 to 6 words to a single word, the precision@1 for the combined dictionary deteriorate slightly but was still over 90%.

Conclusion
We were not able to discover the difference in susceptibility to polysemy between English, Spanish and Mandarin. Hence we went on to explore the effect of induced polysemy on aligning embedding spaces of different languages.

1.3.2 Effect of Induced Polysemy on Cross-lingual Alignment

We aligned cross-lingual embedding spaces where one of them containing induced polysemy by replacing the occurrence of merged words in the training and testing dictionary with the words that they were merged to. Experiment results showed that induced polysemy had minor effects on precision@1 values.

1.4 Word Order

In theory, the order of words in a sentence should not make a difference on the word embeddings trained using word2vec because bag-of-words is used to describe the words within the context window when training the embeddings. Just as a sanity check, we randomly shuffled the word order within each sentence in a corpus before training a new set of embeddings on it. This newly trained embeddings could be aligned perfectly to the original baseline embeddings.
One thing worth consideration is the choice of context window size during training. As smaller windows tend to capture more syntax information and larger window semantic, it could be the case that embeddings of different languages capture different proportion of syntax and semantics information. It can be further investigated whether changing the context window size would lead to a change in the alignment results.

2 Exploiting Bilingual Dictionary by Considering Distance of Cosine Similarity

Inspired by the methods used for analogical reasoning with word vectors, we came up with a new metric for retrieval that would improve the precision of word translation. Previously, when retrieving the translation for a word of interest in the cross-lingual embedding space, we only considered its nearest neighbours, i.e. vectors that maximise cosine similarity with the word vector of interest. However, it occurred to us that we could further exploit the training dictionary apart from just using them for computing the Procrustes matrix. For each word outside the training set, we could evaluate its 'geographical' relationship with the training words in the form of a *cosine similarity vector* where each value within the vector corresponds to the cosine similarity between the word under consideration and one of the training words. In other words, we were using the training vectors as reference coordinates to capture the location of the embedding vector of interest. If a word a_1 was similar to a word b_1 in language L_1, we would expect its translation in language L_2, a_2 to be similar to b_1's translation b_2, as wekk as when two words were disparate in meaning. Hence for a word in the source embedding space, we would expect its counterpart in the target space, which was the word we were trying to retrieve, to have a similar *cosine similarity vector* evaluated on the same set of training words in the target space. This motivated us to add in an additional term, L_2 *distance in cosine similarity vectors*, to the original score function that only considered cosine similarity. Choosing to use L_2 distance instead of L_1 was because we were more concered with large discrepancies in cosine similarities with the 'reference coordinates' since smaller discrepancies were more likely to be caused by noise.

Firstly, we studied how helpful the new information provided by L_2 *distance in cosine similarity vectors* was in helping us predict the correctness of retrieval when we were retrieving translation by (1) only maximising cosine similarity or (2) only minimising L_2 distance in cosine similarity vector (shown in Figure 2). These plots could also be useful in estimating the confidence level of a prediction given the retrieval metric and the values of *cosine similarity* and L_2 *distance in cosine similarity vectors*.

We also plotted the distribution of correctly and wrongly retrieved translations using only the cosine similarity metric (Figure 3).

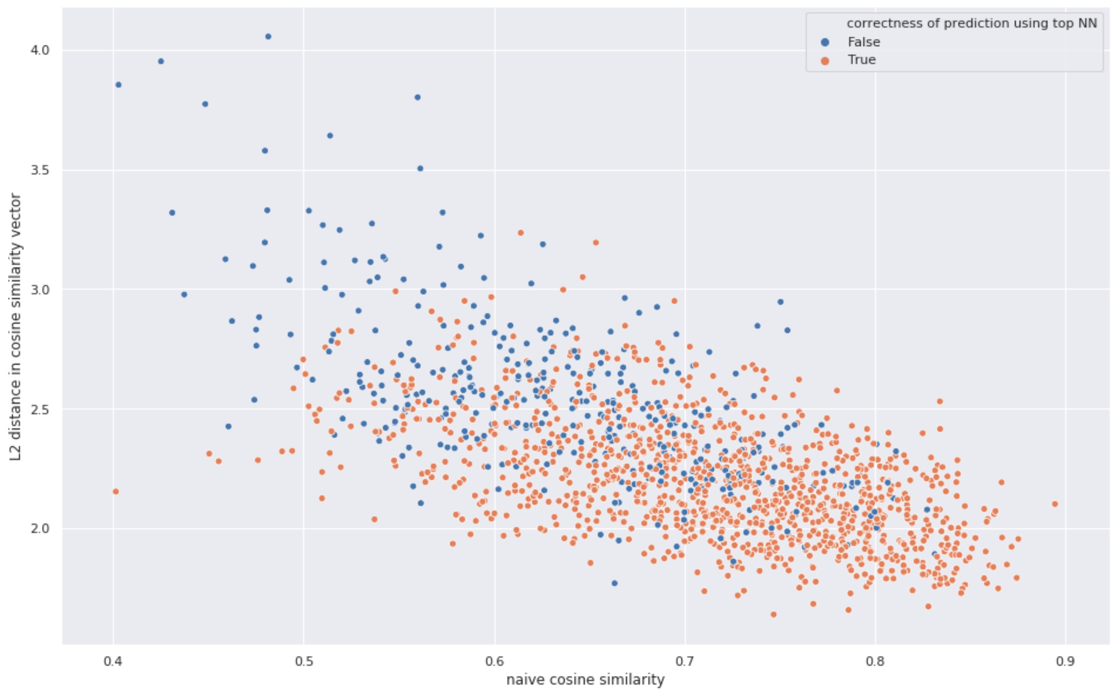
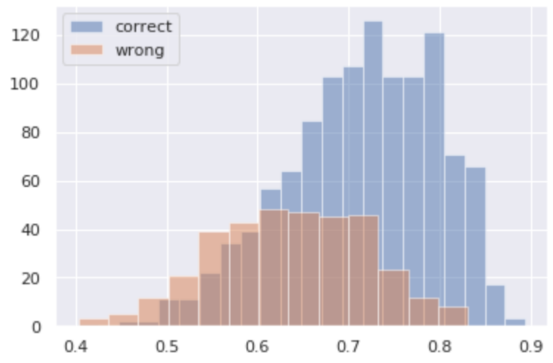
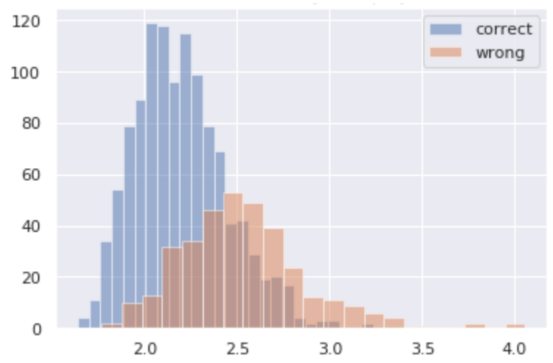


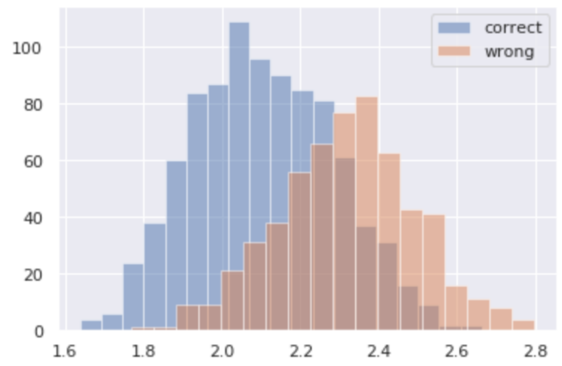
Figure 3: Distribution of correctly and wrongly retrieved translations



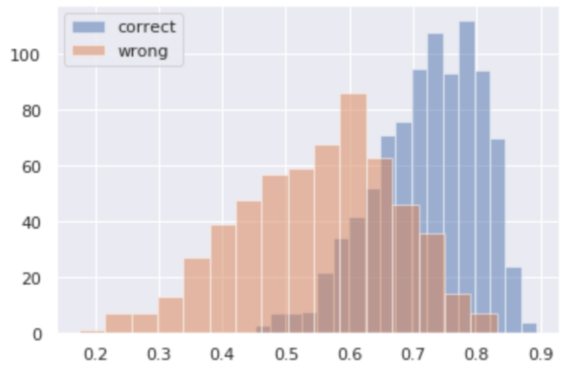
(a) retrieved using only cosine similarity, x-axis shows cosine similarity



(b) retrieved using only cosine similarity, x-axis shows L_2 distance in cosine similarity vectors



(c) retrieved using only L_2 distance in cosine similarity vectors, x-axis shows cosine similarity



(d) retrieved using only L_2 distance in cosine similarity vectors, x-axis shows L_2 distance in cosine similarity vectors

Figure 2: Histograms of correct and wrong retrievals under different retrieval metrics with x-axis showing the value of the two metrics

It was slightly discouraging to see the two measures being so linearly correlated, but this did not completely eradicate the potential to predict the correctness or reliability of the prediction given by cosine similarity alone using e.g. support vector machine.
In subsequent experiments. We combined the two measures to form a new score function $score = cosine\ similarity - \alpha *\ L_2\ distance\ in\ cosine\ similarity\ vectors$. The coefficient multiplied with L_2 *distance in cosine similarity vectors* was used to fine-tune the relative weighting of the two terms. It would take a long time to rank the new score for all words in the target space, hence we only considered the top 100 nearest neightbours as candidate words. Despite this omission, this new score metric was able to improve the precision@1 value by around 1% for all of en-es, en-fr, en-de and en-ru and the best results were often obtained when $\alpha = 0.5$. Note that it is important to normalise *cosine similarity* and L_2 *distance in cosine similarity vectors* term across all the candidate target words before combining the two terms linearly to form the final score. Also, we only used the one-to-one pairs in the training set as the 'reference coordinates' for computing L_2 *distance in cosine similarity vectors* and a potential improvement could be to pick the 'reference coordinates' with more care. For example, we could try to handpick words that cover various domains and categories.
Despite the stable improvement over the baseline retrieval method, our method had the main limitation of not being able to yield an representation for out-of-vocabulary (OOV) words in the projected space. We attempted to incorporate the 'geological information' into the projection matrix and considered jointly optimize the L_2 distance and the 'geological' resemblance between the aligned source space and the target space. However, we were not able to formulate a plausible loss term to describe the latter.
Another primitive thought we had was to make local refinements according to 'geological information'. For example, for each OOV word, we could compute a local projection matrix that minimises the L_2 *distance in cosine similarity vectors* of its nearest neighbours in the training set. To elaborate on this, the algorithm takes in an embedding vector in the source space x and finds its n nearest neighbours in the training set $v_1, v_2, v_3 ... v_n$. These vectors are stacked vertically to form matrix V, with its counterpart in target space being U. The global Procrustes matrix is W_{pro} and v is projected to be $V^{'} = W_{pro}^T V$. For a set of 'reference coordinate' vectors, S_{src} in source space and S_{tgt} in target space, we would want to minimise $\left\| (W_{ref}^T V) S_{src}^T - U S_{tgt}^T \right\|$ where W_{ref} is the local refinement projection matrix. The final refined projection of the source vector of interest would be $W_{ref}^T W_{pro}^T x$.

References

[1] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[2] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," *arXiv preprint arXiv:1710.04087*, 2017.