Notes: “Word translation without parallel data”

Sun

July 27, 2020

1 Summary:

1. Assuming words are independent manifestations of the same event in different languages, and assuming both languages have words that mean the same thing, we should be able to “align” the word embeddings and translate.

2. Using embeddings trained monolingually, using 0 known matching word pairs, the paper presents a method to translate between two (or more) languages.

1.1 High level:

Figure (1) provides a very good idea of the overall idea of the paper. In a nutshell, theoretically, all languages should be equally powerful, and a language should differ from another in nothing but words and grammar. This obviously isn’t true, but its a nice simplification that isn’t very wrong.

In a perfect world, we could merely train two models on separate corpora and the embeddings would already be the same, but this, once again, isn’t actually true. At a minimum, you end up in different local minima, which most likely means different embeddings as well.

So what is there to do? The authors claim, and empirically back up, that you can simply learn a mapping between the two languages.

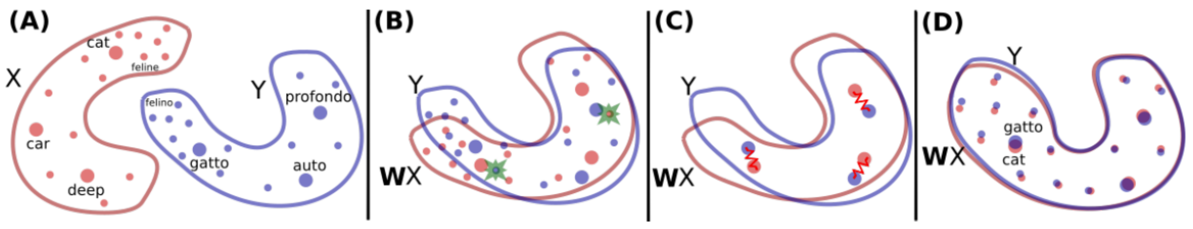

Figure 1: Toy illustration of the method. (A) There are two distributions of word embeddings, English words in red denoted by X and Italian words in blue denoted by Y, which we want to align/translate. Each dot represents a word in that space. The size of the dot is proportional to the frequency of the words in the training corpus of that language. **(B)** Using adversarial learning, we learn a rotation matrix W which roughly aligns the two distributions. The green stars are randomly selected words that are fed to the discriminator to determine whether the two word embeddings come from the same distribution. **(C)** The mapping W is further refined via Procrustes. This method uses frequent words aligned by the previous step as anchor points, and minimizes an energy function that corresponds to a spring system between anchor points. The refined mapping is then used to map all words in the dictionary. **(D)** Finally, we translate by using the mapping W and a distance metric, dubbed CSLS, that expands the space where there is high density of points (like the area around the word “cat”), so that “hubs” (like the word “cat”) become less close to other word vectors than they would otherwise (compare to the same region in panel (A)).

1.2 Low level:

Some background is required to understand why this paper did what it did:

1.2.1 Mikolov et al.:

Before this paper, in 2013, there was a paper that also did this; they learned a linear mapping W between two languages using 5000 known word pairs, such that $W^* = \text{argmin}_{W \in M_d(\mathbb{R})} ||WX - Y||_F$, where F denotes the Frobenius norm, d is the dimension of the embedding, $M_d(\mathbb{R})$ is the space of $d \cdot d$ matrices of real numbers, and X and Y are aligned matrices of size $d \cdot n$ containing the embeddings of the parallel words. The translation of any source word is defined as the most similar vector in the target vocabulary after the transformation is applied: $t = \text{argmax}_t \cos(Wx_s, y_t)$. They found a more complex mapping, such as a multi-layer neural network, did not improve translation.

1.2.2 Xing et al.:

A second paper, in 2015, found that enforcing an orthogonality constraint on W improved translation. When this constrained is applied, it actually changes the problem to one that has a closed form solution.

Procrustes problem: Named after a bandit from Greek mythology who made victims fit his bed by either stretching their limbs, or cutting them off, the Orthogonal Procrustes problem is a problem where one is given two matrices, A and B, and asked to find an orthogonal matrix R which most closely maps A to B. Exactly, $R = \text{argmin}_W ||WA - B||_F$, subject to $WW^T - I = 0$. The solution of this is findable in closed form using some linear algebra magic I don't have intuition for.

1.3 The actual paper:

1.3.1 Adveserial training:

This paper uses an adveserial framework to learn W without any supervision. Given two languages, $\mathcal{X}$ and $\mathcal{Y}$, a model is trained to optimize two terms: $\mathcal{L}_D$ and $\mathcal{L}_W$.

$\mathcal{L}_D$: the discriminator must maximize the probability it predicts whether an embedding is "true" or if the embedding has been mapped from another languages.

$\mathcal{L}_W$: the mapping must maximize the probability the discriminator predicts the origin incorrectly.

This resulting matrix, W, gives good results with just adversarial training, but there are some issues, namely: the generator tries to 'align' all words without regard to their frequencies. Rare words have embeddings that are less well-defined empirically, ergo focusing on the rare words could result in sub-optimal performance.

To refine W, they distill the dictionaries presented. They consider only the most frequent words, and only retain word pairs that have high overlap in nearest neighbors. They then use this as the matching word pair dictionary (similar to Xing et al.) and solve the Procrustes problem.

1.3.2 Cross-domain similarity scaling (CSLS):

Given a good mapping between $\mathcal{X}$ and $\mathcal{Y}$, if x_n has y_m as a near neighbor in language $\mathcal{Y}$, y_m should have x_n as a near neighbor in language $\mathcal{X}$. However, nearest-neighbors is not a symmetric operation, and empirically, this is not true. There are many 'hubs' that are the nearest neighbor of many, while there are 'outliers' that are nearest neighbors of none.

To get better word translations, the authors propose CSLS: a 'dynamic' method of approaching cosine-similarity that expands space around hubs and shrinks it around outliers.

We denote the neighborhood (in the target language) around a mapped source embedding x_s as $\mathcal{N}_T(Wx_s)$, and the neighborhood around y_t as $\mathcal{N}_S(y_t)$. We denote the average cosine similarity of an embedding to its' neighborhood as $r_T(Wx_s) = \frac{1}{K}\sum_{y_t \in \mathcal{N}_T(Wx_s)} \cos(Wx_s, y_t)$, and similarly for $r_S(y_t)$.

We then consider the similarity between two embedding, one in the source language, one in the target, as:

$$\text{CSLS}(Wx_s, y_t) = 2\cos(Wx_s, y_t) - r_T(Wx_s) - r_S(y_t) \tag{1}$$

Again, intuitively, this increases the similarity of isolated embeddings, while reducing the similarity of hubs.

1.3.3 More intuition:

Since meaning is independent of language, we should be able to generate embeddings for two languages, and have them align automatically. This isn't the case, and in fact, isn't even the case for one language. If we train two seperate models to generate embeddings for monolingual data, we will get different embeddings each time.

In the case of word2vec/Glove, this is partially due to the fact that if R is a rotation matrix, and W is an embedding that minimizes the loss, then RW does as well. Ergo, there is no single solution, but a manifold.

So, there should be a rotation matrix that will map one set of our monolingual embeddings to the other. And since meaning is independant of language, this should be true for embeddings in different languages too.