Report on Word Embeddings

Emma COVILI, Salomé PAPEREUX, Éloïse ZAGHRINI
— DeepL —

November 17, 2021

1. Introduction

Our goal is to create a translator between french and english, and vice-versa. In order to do that, we are going to use word embeddings. This is a vector representation of a word. It allows words that have approximately the same meaning to have vectors with approximately the same values. We used the embeddings given in the MUSE repository [1], trained on the Wikipedia Corpora. There exists two ways to build a translator : the supervised and the unsupervised methods. In the first one, we use a bilingual dictionary to learn a transformation and in the second we don't have one and we try to learn the transformation with Generative Adversarial Networks.

2. Methods

(a) The supervised approach

Firstly, we implemented the supervised approach. As we knew we would need to code two versions of this, we decided to create a class `Initializer` with built-in methods such as `RMSE`, `accuracy` or `predict`. This class is initialized with data from the test sets, it has others attributes such as the matrix W which is initialized randomly, and a list of the 7 nearest words in the vocabulary for each word and a list of accuracies according to the number of neighbors taken into account.

i. Linear transform

The idea is to consider a set of pair of words and their associated vector representations $\{x_i, z_i\}_{i=1}^{n}$, where $x_i \in R^{d_1}$ is the distributed representation of word i in the source language, and $z_i \in R^{d_2}$ is the vector representation of its translation. In our case, $d_1 = d_2 = d$ and $d = 300$. It is our goal to find a transformation matrix W such that Wx_i approximates z_i. In practice, W can be learned by the following optimization problem:

$$\min_W \sum_{i=1}^{n} \|Wx_i - z_i\|^2 \tag{1}$$

which we solve with a linear transformation.

We implemented a class `LinearTransform` that builds onto the `Initializer` one, in order to compute W. This class computes the Ordinary Least Squares regularized solution, known as

$$W = (XX^T - \lambda Id)X^T Z \tag{2}$$

Finally, we compute the product $\hat{Z} = WX$. The mapping not being exact, we try to find for each embedding $\hat{z_i}$, the closest target embedding among those of the test set. To determine the distance between two embeddings, we used several distances: the ordinary vector norm in a first time, and the cosine similarity in a second time. For all the final results of the project, we used the distance cosine similarity, because it gives better results and is faster to compute. The details of this choice are justified in the results section.

ii. Orthogonal transform

We solve this problem, like in the paper Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation[2], by using the norm distance in the transform learning :

$$\min_W \sum_{i} ||Wx_i - z_i||^2 \tag{3}$$

To solve the problem, we first consider the case where the dimensions of the source and target vector spaces are the same. In this case, the normalization constraint on word vectors can be satisfied by constraining W as an orthogonal matrix, which turns the unconstrained problem 3 to a constrained optimization problem. A general solver such as SQP can be used to solve the problem. However, we seek a simple approximation in this work. Firstly, solve 3 by gradient descendant without considering any constraint. A simple calculation shows that the gradient is as follows:

$$\nabla W = 2X^T(XW - Z) \tag{4}$$

and the update rule is simply given by:

$$W = W + \alpha \nabla W \tag{5}$$

where α is the learning rate. After the update, W is orthogonalized by solving the following constrained quadratic problem:

$$\min_{\bar{W}} \|W - \bar{W}\| \;\; s.t. \;\; \bar{W}^T\bar{W} = Id \tag{6}$$

One can show that this problem can be solved by taking the singular value decomposition (SVD) of W and replacing the singular values to ones.

(b) The unsupervised approach

Secondly, we implemented the unsupervised approach as described in the paper Word Translation Without Parallel Data [3]. In this approach, we suppose that we don't know the pair of corresponding words from source embeddings to target embeddings. This method then relies on the adversarial training of two models, a generator (or mapping) and a discriminator.

Let $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Y} = \{y_1, ..., y_m\}$ be two sets of n and m word embeddings coming from a source and a target language respectively. The first model is then the mapping matrix W that solves the following problem :

$$\underset{W \in M_d(\mathbb{R})}{argmin} \, ||WX - Y|| \tag{7}$$

By adding an orthogonalization constraint on W, as in the supervised methods, we increase the accuracy. At each step, we therefore perform the following update to keep that constraint :

$$W \leftarrow (1 + \beta)W - \beta \left(WW^T\right) W \tag{8}$$

A second model called a discriminator is then trained to learn to differentiate the origin of the embeddings it receives as input. It should classify the true target embeddings Y as 1 and the generated WX embeddings as 0.
The mapping is trained to fool the discriminator, which means that its goal is to make the discriminator output 1 for generated embeddings WX. The training then consists of an alternate stochastic gradient update of each model with their loss being opposed. As the discriminator becomes stronger, so does the mapping which outputs WX. All the generated embeddings become closer to belonging to the target latent space, and the similarities between languages embedding spaces make that they will fall close to the real translations according to cosine similarity.

We found our best results with a discriminator model of 3 hidden blocks of a fully connected layer of 2048, a dropout and a leaky ReLu activation, compared to two in the paper. We also changed the learning method for the mapping model by only propagating the loss on generated data, making the generator learn faster. This translates to propagating the following losses :

$$\mathcal{L}_D(Discriminator|W) = -\frac{1}{n}\sum_{i=1}^{n} \log P(source = 0|Wx_i) - \frac{1}{m}\sum_{i=1}^{m} \log P(source = 1|y_i) \quad (9)$$

$$\mathcal{L}_W(W|Discriminator) = -\frac{1}{n}\sum_{i=1}^{n} \log P(source = 1|Wx_i) \quad (10)$$

After enough iterations, we then use a Procrustes refinement method. This method relies on taking the embeddings that W outputs and the target embeddings of the most frequent words, and looking at the closest pairs according to cosine similarity. We then reduce the distance between those words, to try to match perfectly the target embedding space. This is done using Procrustes Algorithm, which is relies on updated W iteratively using singular value decomposition.

3. Results

Our main objective is to build models that best translate a given language into another language, word by word. More precisely, we need to maximize the global predictive accuracy of these models, i.e. maximize the amount of correct translation from an initial language into a given target language. The different languages on which we have tested our models are : English - French, French - English, French - Italian, Italian - French, English - Greek, Greek - English and English-Chinese. Another objective is to minimize the execution complexity of the algorithms. Associating each source embedding prediction with the nearest target embedding and then with the associated target word is a time consuming task.

(a) Supervised approach : linear VS orthogonal transform

i. Norm distance transform

In the cases where we used the ordinary vector norm, the orthogonal transformation was much more efficient than the linear transformation. This was the case in terms of accuracy: 37%, 84%, 46% for the orthogonal VS 22%, 78%, 20% for the linear approach, for the English-French, French-Italian and Greek-English translations respectively. On the other hand, the linear approach is 450 times faster than the orthogonal approach: 0.1 seconds versus 45 seconds. However the time to predict the translations, and the accuracy calculations are very long, up to several minutes.

ii. Cosine Similarity distance

In the case where we used the cosine similarity, the prediction and accuracy computation time has greatly improved, due to the ease of use of the `cosine similarity` method of the `sklearn` library. The results are almost instantaneous. Even better, the accuracy improves even more for the linear and orthogonal transformation methods. The accuracy is 42%, 88%, 51% for the linear VS 42%, 90%, 54% for the orthogonal, for the English-French, French-Italian and Greek-English translations respectively. The accuracy of the two models are equivalent with this distance.

(b) Influence of language on accuracy

To compensate for the fact that the W-mapping is not exact, and the accuracy of the precision is slightly biased, the nearest 7 neighbors of each translation prediction were calculated. We first ran tests with 10-NN, but the accuracy results plateau from 8-NN onwards. Here are the translation results for three of the 7 translation pairs performed.

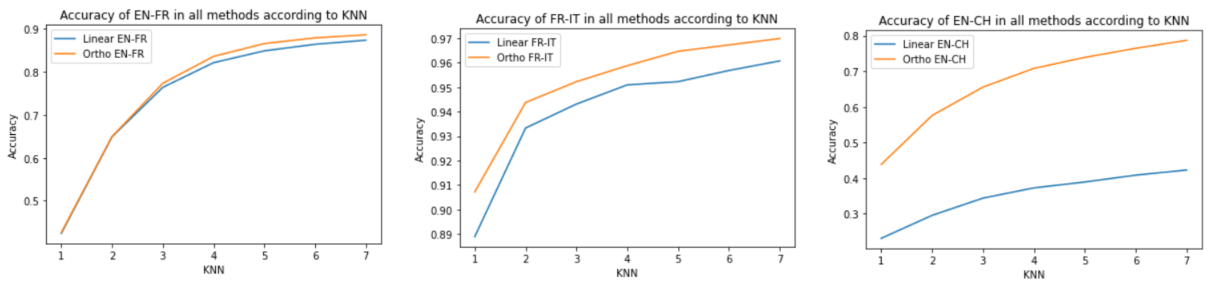

Fig 1 : Accuracy of multiple language translations according to the number of nearest neighbors

First of all, we notice that the accuracy with 1-NN starts much better for the French-Italian translation. This is surely due to the proximity of the languages, coming from Latin. Then, the English-French translation is ahead of the English-Chinese translation. Such a discrepancy can be explained by the difference in the alphabet, but also the grammar and turns of phrase, which are completely different. In the end, for 7-NN, the French-Italian translation is ahead of the others with an accuracy of 95%, with 90% for English-French and 80% for English-Chinese. Moreover, the procrustean method asserts its efficiency and stability compared to linear transformation by widening the prediction gap with a language such as Chinese.
Here is a representative graph of the translation efficiency with the supervised methods.

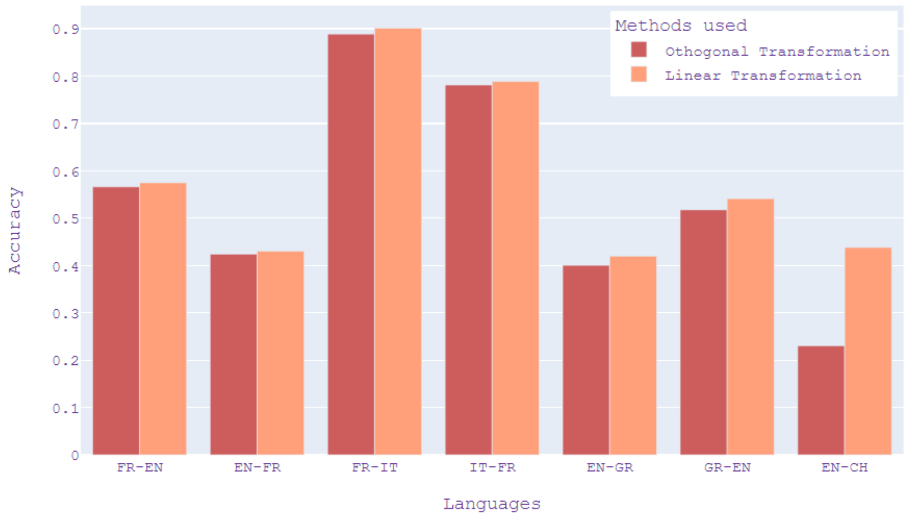

Fig 2 : Accuracy of multiple language translations according to supervised methods

In the appendix appears the summary table of all accuracies for all language translations with supervised methods, for 1 and 7 nearest neighbors.

(c) Unsupervised Approach

When implementing the generative adversarial network, we found that a lot of different parameters and choices can affect the result drastically. Because the two models need to be trained at the same time, it's easy for one to overtake the other if the parameters are not good. Contrary to our first belief, training the discriminator in a half supervised way only

on batches of source words and their translations lead to very poor results. We increased the accuracy drastically when using completely different batches for X and Y. We also ended up not using weight decay, as it lead to a slower and weaker training for us. As mentioned previously, focusing the loss of the generator only on the discriminator's guesses on generated data also improved our results a lot.
The refinement steps proved to be very useful and to improve the accuracy by +10-20% in our experiments. However, we noticed that choosing too many embeddings to compute the closest one on which to perform the Procrustes refinement could also lead to a decrease in accuracy. We found our best results by focusing on the 10k most frequent words.
Compared to the supervised approach, we could confirm that this method often leads to better results according to the 1 nearest neighbors accuracy (see figure 3). In cases where the languages are too different, such as with Greek and english, it can however struggle to outperform the supervised way.

	fr-en	en-fr	fr-it	it-fr	en-el	el-en
Supervised methods						
Linear - 1-NN	0,57	0,42	0,89	0,78	0,40	0,51
Procrustes - 1-NN	0,57	0,43	**0,90**	0,79	0,42	**0,54**
Unsupervised methods						
GAN - 1-NN	0,66	0,70	0,68	0,76	0,20	0,11
GAN & refinement - 1-NN	**0,77**	**0,79**	0,79	**0,85**	**0,43**	0,21

Fig 3 : Word translation accuracies according to methods and languages

(d) Hubness problem

During the development of the supervised approach, we faced the hubness problem. For example, the words "problématique" and "sûrement" are often the translated words in the English-French translation. We were inspired by the solution provided in [3] to overcome this problem. In the same way as before, we have, for each source word, predicted the translation with the closest word (we call it predicted word). From this predicted word, we checked, in reverse, that the source word is in the k nearest neighbors of the predicted word. If it is, then we assign this word as the translation of the source word. If this word doesn't exist after a fixed number of iterations, then we take the initial predictions of the source word. We repeat this to get the 7 nearest neighbors of the source word. And this for each word of the test set. This method is only effective in the context of English-French translation. We gain 37% accuracy, from 42% to 57%, with $k = 7$. In other language pairs, the translation accuracy is reduced from 5% to 10%. On the other hand, this method is obviously slower.

4. Conclusion

This project allowed us to understand NLP through the embedding spaces and to code our first generative adversarial model. We were able to visualize the properties of the vector representation of languages, and through our experiments we were able to find a very efficient model for translation. We have confirmed the results of our various researches on the subject, and obtained impressive accuracies for unsupervised models. This project was for our group a first introduction to GAN and NLP, and gave us an inspiring insight into the possibilities of such models. Even if these models are efficient, they would not be enough to make translations as accurate as those presented by DeepL or Google Translation, for example. Especially since the translation is done word by word and does not take into account the context, which has a huge influence on the translation.

Bibliography

[1] FAIR, "Muse : Multilingual unsupervised and supervised embeddings." `https://github.com/facebookresearch/MUSE` (2018).

[2] Xing, C., Wang, D., Liu, C., and Lin, Y., "Normalized word embedding and orthogonal transform for bilingual word translation," 1006–1011 (2015).

[3] Lample, G., Conneau, A., Ranzato, M., Denoyer, L., and Jégou, H., "Word translation without parallel data," (2018).

APPENDIX

	fr-en	en-fr	fr-it	it-fr	en-el	el-en	en-ch
Supervised methods							
Linear - 1-NN	0,57	0,42	0,89	0,78	0,40	0,51	0,23
Linear - 7-NN	0,86	0,87	0,96	0,94	0,76	0,79	0,42
Procrustes - 1-NN	0,57	0,43	0,90	0,79	0,42	0,54	0,43
Procrustes - 7-NN	**0,87**	**0,89**	**0,97**	**0,96**	**0,78**	**0,82**	**0,79**

Word translation accuracies according to method and nearest neighbors