# Bilingual Translation with Word Embeddings

Matthieu Rolland, Ryan Belkhir, Germain Bregeon

Data Science Project

Master IASD, Université Paris Dauphine-PSL

.

**Keywords:** Word Embeddings, MUSE, Monolingual Translation, GAN

## ABSTRACT

In this paper we summarize our attempts to use high level representations of words such as word embeddings to build a monolingual word translator. We present several methods from different research papers to train a linear model in supervised and unsupervised settings. Finally we display and comment our results and algorithm's performances and provide ways of future improvement.

## Contents

# 1. INTRODUCTION

## 1.1 Recall on word embedding

In natural language processing (NLP), Word embedding is the high level representation of words in a vector space of variable dimension (300 in our case) for text analysis, typically in the form of real-valued vectors. Each vector (corresponding to a word) encodes the meaning of a word based on it's context (adjacent words in texts). This main assumption is based on the distributional hypothesis which states that words that appears in similar contexts are close in meaning. Therefore, vector that are close in context (meaning) have close representation in the vector space of embeddings (see Figure 1). The vector space of embeddings has also good properties:

- It is geometric: $v(king) - v(man) + v(woman) = v(queen)$
- Closeness between two vectors can be easily evaluated using Cosine Similarity : $cos(\theta) = \frac{\vec{u}\vec{v}}{\|\vec{u}\|\|\vec{v}\|}$

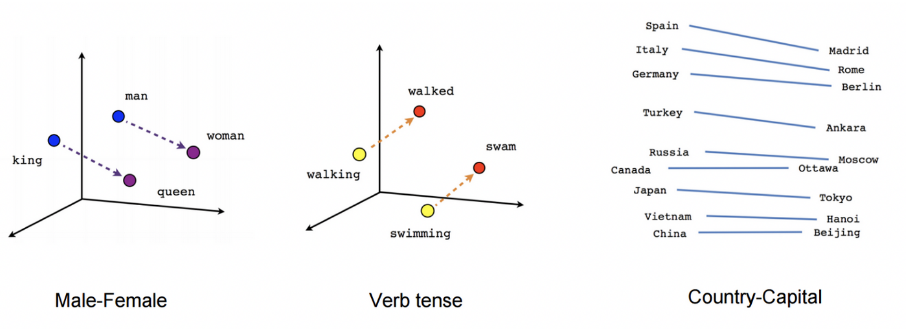

Figure 1. Example of word embeddings in dimension 3

## 1.2 Problem definition

The goal of this project was to learn and become familiar with high level representations (embeddings) of things in problem solving tasks as well as experimenting the geometric nature of deep neural networks. Our instance of this project is to build an efficient word translator for English to French and French to English based on the use of word embeddings. The base principle is to exploit linear transformations and rotations of vector spaces to translate a word. This is equivalent to learn **W** with $X$ (source language word embeddings) and $Y$ (target language word embeddings) such that:

$$Y = \mathbf{W}X$$

Provided 200.000 word embeddings in dimension 300 from the FastText library, we'll be using two different approaches:

- A Supervised approach using a parallel corpus of translations
- An Unsupervised approach using GANs (generative adversarial networks)

Still, it shall not be forgotten that the quality of a translation is based on the quality of the embeddings to generalize to many subjects, therefore the types of text that are used to build the embeddings shall not be too specific.

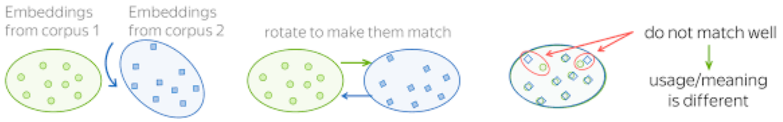

Figure 2. Linear rotation and translation to align 2 vector spaces

## 2. OUR PROCESS AND APPROACHES

**We first started** by getting familiar with the concept of word embedding and supervised methods. We started by reading different papers proposing various methods of regression, in particular:

- An ordinary least squares regression from ”Exploiting Similarities among Languages for Machine Translation”[1]
- An orthogonal transform applied on ordinary least squares from ”Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation”[2]
- A Procrustes method applied to ordinary least squares with Frobrenius norm from ”Word Translation without parallel Data”[3]

**We then followed** by getting familiar with the concept of unsupervised methods and GANs. We maily used the GAN method and tuning from:

- ”Word Translation without parallel Data”[3]

**Next** we have used these methods to study the transitivity between several languages. To see if the use of an intermediate language could improve the results on translation between two other languages.

**Finally** we decided to apply a K-Means algorithm to divide a language into different vocabularies and test if training by vocabulary could be more efficient than global learning.

## 2.1 Supervised Approach

The ordinary least squares regression from ”Exploiting Similarities among Languages for Machine Translation”[1] learns a linear transform from the source language to the target language. The objective function is as follows:

$$\min_{W} \sum_{i} \|Wx_i - z_i\|_2^2 \quad \text{or} \quad \min_{W} \|WX - Z\|_F^2 \tag{1}$$

where W is the projection matrix to be learned, and $x_i$ and $y_i$ are word embeddings vectors in the source and target language respectively. Once W is learned, we can easily translate any word by finding the nearest neighbour vector of its transformation by W using the cosine similarity metric.

### 2.1.1 Ordinary Least Squares

In this section, we implement the ordinary least squares regression (1) and seek to compute a baseline of results to evaluate the translation performances of our other methods. We chose two different minimization techniques in this case namely Gradient Descent (GD) and closed form least squares solution calculation to compare in our case (not possible with huge datasets). See results in Figure 3.

### 2.1.2 Ordinary Least Squares with Orthogonal transformation

In this section, we seek to implement the ordinary least squares regression in (1) adding a normalization of source and target word embeddings and an orthogonality constraint on W to counterbalance certain inconsistencies as specified in ”Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation”.[2] Nevertheless, the iterative solution of this paper only led us to the Procrustes's result in very few iterations and the computation time was very much higher than our other methods. Therefore we only applied the orthogonalisation of W at each GD iteration to optimize its performances. See results in Figure 3.

### 2.1.3 Procrustes

In this section, we seek to find the closed form solution of the ordinary least squares regression in (1) adding an orthogonality constraint on W. In that case, the equation (1) boils down to the Procrustes problem, which advantageously offers a closed form solution obtained from the singular value decomposition (SVD) such as

$$W^\star = \underset{W \in O_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Z\|_{\mathrm{F}}^2 = UV^T, \text{ with } U\Sigma V^T = \mathrm{SVD}\left(ZX^T\right)$$

### 2.1.4 Supervised learning results

In Figure 3 we plot and compare our supervised accuracy results for all previously mentioned methods (en-fr/fr-en and en-ru/ru-en translations).

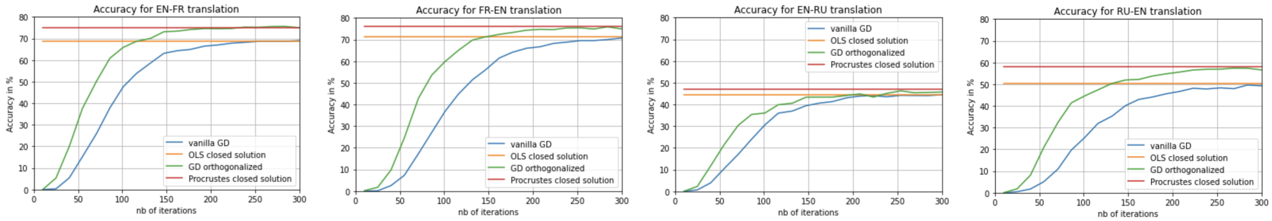

Figure 3. Supervised Learning accuracy results for EN-FR/FR-EN and EN-RU/RU-EN translation using all methods

| | en-fr | fr-en | en-it | en-es | en-in | es-it | pl-en |
|---|---|---|---|---|---|---|---|
| *Supervised methods* | | | | | | | |
| GD Ortho - NN | 73.47 | 75.43 | 66.93 | 75.54 | 49.0 | 77.12 | 54.8 |
| Procrustes - NN | 74.93 | 76.07 | 68.93 | 77.4 | 51.57 | 78.67 | 57.93 |

Table 1. Nearest Neighbor (NN) Accuracy results in between several different languages in supervised training

Hence, it is fair to say that linear models are pretty efficient in regards to their simplicity and quickness of implementation for such a task.

## 2.2 Unsupervised Approach

In the unsupervised approach we focused on ”Word Translation without parallel Data”[3] paper which learns a matrix $W$ to align 2 languages embedding vector spaces as follows:

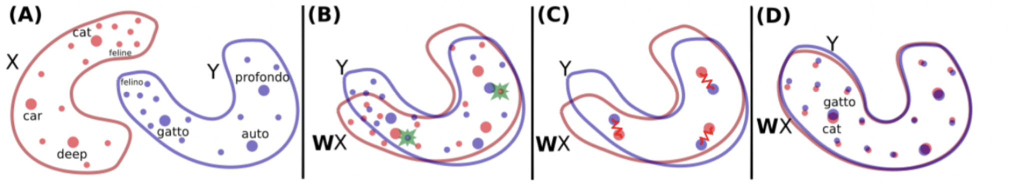

Figure 4. Linear translation and rotation of two embedding vector space

- (A) There are two distributions of word embeddings denoted by X and Y, which we want to align/translate (each dot represents a word in that space)
- (B) Using adversarial learning, we learn a rotation matrix W which roughly aligns the two distributions. We choose 2 words randomly (green stars) 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. 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.

### 2.2.1 GAN

In order to learn $W$, the approach of using a GAN is chosen. The model is trained in a two-player game configuration. We train conjointly a discriminator that discriminates between elements randomly sampled from our predictions $(WX)$ and $Y$. The Generator is trained to prevent the discriminator from making accurate predictions. As a result, the discriminator aims at maximizing its ability to identify the origin of an embedding, and the generator aims at preventing the discriminator from doing so by making $WX$ and $Y$ as similar as possible.

Considering the discriminator parameters as $\theta_D$ the discriminator loss can be written as:

$$\mathcal{L}_D\left(\theta_D \mid W\right)=-\frac{1}{n} \sum_{i=1}^n \log P_{\theta_D}\left(\text { source }=1 \mid W x_i\right)-\frac{1}{m} \sum_{i=1}^m \log P_{\theta_D}\left(\text { source }=0 \mid y_i\right)$$

The Generator loss can be written as:

$$\mathcal{L}_G\left(W \mid \theta_D\right)=-\frac{1}{n} \sum_{i=1}^n \log P_{\theta_D}\left(\text { source }=0 \mid W x_i\right)-\frac{1}{m} \sum_{i=1}^m \log P_{\theta_D}\left(\text { source }=1 \mid y_i\right)$$

### 2.2.2 Procrustes refinement

As described in the paper[3] we performed a refinement procedure in order to compare the results and see if it really allows to gain in performance.
To do this we calculated the distance between each word and selected the words that were mutual nearest-neighbors. We then built a dictionary with these words and applied the procrustes method described above. Each iteration allowing us to obtain a new and more precise dictionary, we could therefore iterate this process several times.

### 2.2.3 A new metric : Cross-Domain Similarity Local Scaling (CSLS)

In addition to a new method, the paper[3] also presented a new metric to find two nearest neighbors : the CSLS. The main idea of this metric is to counter the hubness problem, i.e. the cases where some words are nearest neighbors to many words while others are not nearest neighbors to anyone.

$$CSLS(Wx_s, y_t) = 2\cos(Wx_s, y_t) - r_T(Wx_s) - r_S(y_t)$$

where $\cos(.,.)$ represent the cosine similarity between two words and $r_T(.), r_S(.)$ the mean similarity of a word to its neighborhood.
Intuitively, this means penalizing words with a lot of nearest-neighbors and valuing those that don't have many.

## 2.3 Our results

As we can see in Table 2, languages with have a common background (such as Spanish and Italian here) have overall better results than the translation between other languages. We can also note than in the case where the languages do not have a common background, the refinement procedure in GAN increases drastically the results (for example, in en-in translation, the refinement increases the accuracy by 34.87% when using NN and 31.4% when using CSLS). We can thus conclude, that the refinement procedure in GAN is mandatory when trying to translate between two languages that do not have a common background.

| | en-fr | fr-en | en-it | en-es | en-in | es-it | pl-en |
|---|---|---|---|---|---|---|---|
| *Unsupervised methods, using GAN* | | | | | | | |
| NN | 66.53 | 62.94 | 56.27 | 66.53 | 21.33 | 70.53 | 23.87 |
| NN+refinement | 77.4 | 76.89 | 74.67 | 79.8 | 56.2 | 80.4 | 58.46 |
| CSLS | 72.07 | 68.07 | 63.13 | 69.87 | 31.33 | 73.07 | 35.53 |
| CSLS+refinement | 80.87 | 80.73 | 76.73 | 80.8 | 62.73 | 81.6 | 65.2 |

Table 2. Unsupervised methods accuracy results

# 3. LANGUAGE TRANSITIVITY & VOCABULARY CLUSTERING

## 3.1 Language Transitivity

Our first idea was to test the transitivity between different languages. Even though we had little hope of gaining performance this way, we wanted to see if using an intermediate language could be beneficial. Intuitively we decided to test this method by using intermediate languages that are close or have common roots like French and Italian with Latin.

| | it-fr | fr-en | it-en | it-fr-en |
|---|---|---|---|---|
| GD | 77.3 | 70.8 | 67.0 | 60.2 |
| Procrustes | 81.67 | 74.8 | 70.73 | 64.0 |

Table 3. Italian to English training using French as an intermediate language

We can see that even when using close languages that have good performances like French and Italian. Going through French to translate Italian into English makes us lose in performance despite the fact that the accuracy between Italian/French and French/English are both higher than that of Italian/English.

*NB :* Due to time constraints we have only tested the supervised methods.

## 3.2 Vocabulary Clustering

Our next idea was to use a K-Means algorithm to pre-process the data and divide it into different vocabularies. We then compared the results after global training and after vocabulary training.
To find the best hyperparameter k, we used the elbow and silhouette methods. But we noticed that these methods gave us too large cluster numbers. And that despite some very interesting vocabulary (we had for example a vocabulary composed only of countries, capitals, geographical places...) on which the performances were better than with a global training. Too many vocabularies made the methods too unstable as some vocabularies ended up with too few words and therefore very low results.
We therefore decided to start with a smaller number of vocabulary, $k = 4$:

| | Voc 1 | Voc 2 | Voc 3 | Voc 4 |
|---|---|---|---|---|
| Global training (Procrustes) | 80.4 | 75.29 | 64.97 | 81.49 |
| Vocabulary Training (Procrustes) | 79.9 | 74.41 | 66.88 | 79.17 |

Table 4. Vocabulary vs Global training on English to French translation

As we can see the vocabulary training can have benefits as it is the case in cluster n°3 but also degraded the results compared to the global training.

*NB :* We implemented a KMeans algorithm but for efficiency reasons we then decided to use the one from the Scikit-learn library.

## 4. LEARNING OUTCOMES AND NEXT STEPS

**With this project we could learn about:**

- The importance of latent space representation to identify features in complex data, in particular with neural networks,
- The efficiency of simple linear models in supervised word translation tasks,
- How to design, train and refine Generative Adversarial Networks in a particular context which can be extended to several other tasks,
- Understanding and implementing recent research papers from scratch on our own.

**Given more time, we would:**

- Consider non linear models in supervised word translation tasks and compare with linear models in terms of accuracy, time and implementation complexity. In particular ”Supervised and Nonlinear Alignment of Two Embedding Spaces for Dictionary Induction in Low Resourced Languages”[4]
- Going beyond the unsupervised current paper, we would want to further explore multilingual and sentences translation using the paper ”Unsupervised Multilingual Alignment using Wasserstein Barycenter”.[5]

# REFERENCES

[1] Tomas Mikolov, Quoc V. Le, I. S., "Exploiting similarities among languages for machine translation," (2013).

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

[3] Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H., "Word translation without parallel data," (2017).

[4] Moshtaghi, M., "Supervised and nonlinear alignment of two embedding spaces for dictionary induction in low resourced languages," (2019).

[5] Xin Lian, Kshitij Jain, J. T. P. P. Y. Y., "Unsupervised multilingual alignment using wasserstein barycenter," (2020).

## 5. APPENDIX

**GAN plots:**

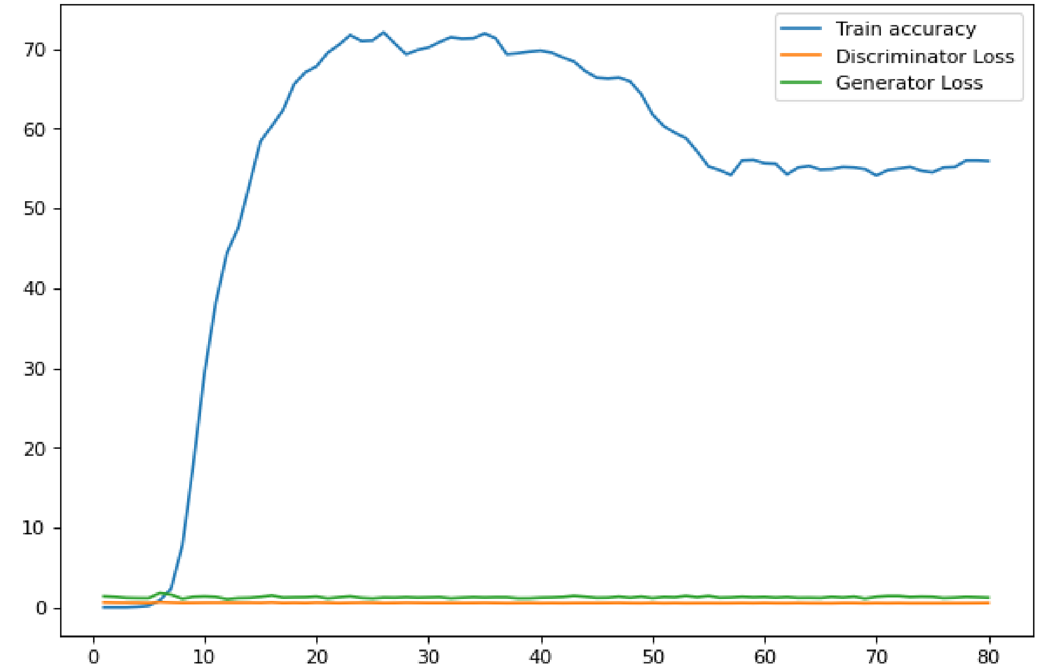

Figure 5. Plot of our GAN training with CSLS on English to French