Machine Translation Using Word Embeddings

Data Science Project
Master IASD

November 19th, 2020

Alejandro Castro Ros, Eloi Massoulié, Hassan El Mansouri Khoudari

Contents

Abstract

Machine Translation has become an important topic of Artificial Intelligence research in the past few years. In this field, there has been a wide variety of approaches; from methods relying on pairs of translated words to independent methods than can be applied to multilingual translation. The purpose of this work is to get familiar with word embeddings, which serve as the basis of machine translation methods, and to make a comparison of several approaches to this interesting Natural Language Processing task.

1 Introduction

Since the beginning of the development of Artificial Intelligence, one of the main objects of research has been human language, giving raise to what is today known as Natural Language Processing (commonly referred as *NLP*). A wide range of Machine Learning techniques has been applied to this purpose, using them for different tasks related with human language: from text classification (Natural Language Understanding, *NLU*) to automatic summarization or text generation (Natural Language Generation, *NLG*).

One of the more important tasks related to NLP is the translation of texts from a source language into one or many target languages (commonly known as Machine Translation). To this effect, many methods employing different techniques and with different degrees of complexity have been proposed.

1.1 Word Embeddings

All of forementioned NLP tasks need to treat human language as a set of mathematical elements (numbers, vectors, matrices, etc.). The mapping between any piece of language (word, sentence, document, etc.) and a corresponding mathematical representation is called an **embedding**. There exist different approaches to infer the mapping of text sequences into a vector space, using multiple techniques such as Latent Semantic Analysis or Neural Networks.

For the purpose of this document we will use pretrained word embeddings for different language coming from *fastText*.

1.2 fastText

fastText is a library designed to generate word embeddings developed by Facebook. Its method is based on the skipgram model. It provides with pretrained word embeddings as well as word to word translation dictionaries for different languages. During our work, we have used the available resources for English, French, Spanish, Arab, Korean and Japanese. All these resources are publicly available under *BSD* License [6].

2 Problem Definition

3 Methods

There are many different approaches to the problem defined in the previous section. They can be divided into two groups depending on the necessity of having a predefined set of word translations from the source to the target language. In case these translations are needed, we would be addressing the problem with a Supervised Learning method. On the other hand, when there is no need of previous translations, the method would belong to the group of Unsupervised Learning techniques, which are usually very interesting for the ease of generalization.

3.1 Supervised Learning

In this section we will explain the methods we have applied to address the task of finding a mapping from the source word embedding to the target one with a set of predefined translations between

them.

3.1.1 Overview of the methods tested

We consider that both the source language and the target language already have an adequate word embedding, denoted by the respective vector spaces X and Z. The techniques we have adopted involve normalising the vectors in these embedding spaces, so that everything is on the surface of a sphere - from there, the transformation from one language to the next should be adequately approximated with a rotation. The task is therefore, given a loss function l, to find an orthogonal matrix W which minimises $l(Wx, z)$ for all source-target correspondences $(x, z) \in X \times Z$.

Looking for an orthogonal matrix means that the context of the problem is no longer a vector space, and gradient descent alone cannot find the minimizer. The approach favoured by similar work seems to be separation of the two constraints: finding any optimal matrix W, then computing an orthogonal matrix $\tilde{W}$ that approximates W. This can be done by expressing W as its singular value decomposition and setting every singular value to one (in other words, for the decomposition $W = UsV$ we can express an adequate rotation as $\tilde{W} = UV$.

For the loss function, two options were explored : the standard quadratic loss

$$l(X, Z) = \sum_i \|Wx_i - z_i\|^2 \tag{1}$$

and the cosine distance, which in a normalized space is simply a dot product

$$l(X, Z) = -\sum_i (Wx_i)^T z_i \tag{2}$$

(since the cosine distance varies between 0 and 1 with a value of 1 for identical vectors, this cost must rather be maximized, hence the negative sign).

A third method was explored as well, it is the Procrustean optimization problem. It is a closed form solution for the orthogonal matrix W that minimizes the following quantity according to the Frobenius norm:

$$\tilde{W} = \underset{W \in O_n}{\arg\min} \|WX - Z\|_{\mathcal{F}} = U \cdot V^T \quad \text{with} \quad U\Sigma V^T = SVD(ZX^T) \tag{3}$$

This method serves us as a baseline for comparing with other methods.

3.1.2 Experimental results

For each of the iterative methods exposed, we have run trainings with different hyperparameters in order to obtain the best results (since the Procrustes is a closed-form solution there are no hyperparameters to tune). We have changed the number of iterations (epochs), the learning rate and the initialization of the mapping W at the beginning of the training in order to control the best setting and optimize our translation system. As an example, for the way we initialize the matrix W for the iterative methods described by Equations (1) and (2), we have compared between using an Identity matrix, a matrix composed only of 0s, a matrix composed of 1s and a randomly initialized matrix. For the quadratic loss optimization, the mean cosine similarity between the target embedding and the transformation of the source embedding for each of these initialization

techniques can be observed in Figure 1. While the differences are not very significant, we have accordingly chosen to work with the initialization as a matrix with only zero values.

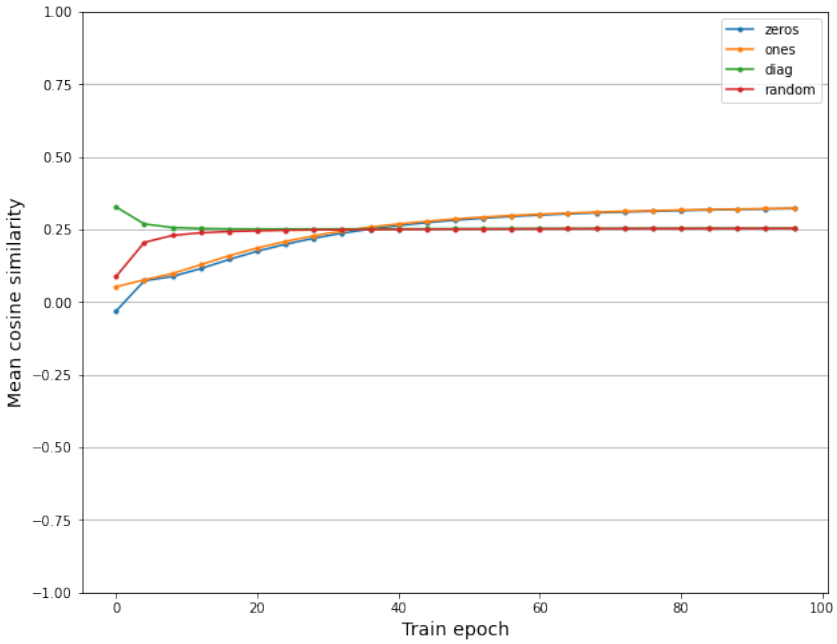

Figure 1: Comparison between different initialization functions for W.

In the same way, the learning rate was found to reach an optimal value for 0.05.

As a illustrative example, we are going to analyze the translation of the word "sit" into French using the three methods described by Equations (2)-(3). For obtaining the translation, we transform the source embedding vector via the matrix W and look to the closest words in the target space (k-NN). The results obtained are represented in Table 3.1.2.

	Linear Loss	Quadratic Loss	Procrustes
1	remit (0.185)	pivoter (0.320)	**asseoir** (0.502)
2	aïeux (0.180)	glisser (0.307)	dormir (0.499)
3	battit (0.179)	hall (0.305)	**assis** (0.492)
4	retourna (0.176)	**asseoir** (0.305)	reposer(0.485)

Table 1: Top 4 closest words to *sit* for different supervised methods.

We can see that the Procrustes method obtains better result that the iterative ones. It is also important to notice the low score obtained by the nearest neighbours in the iterative method based on the Linear Loss.

3.1.3 Language Comparison

In order to analyze the generalization of the methods aforementioned, we have selected different languages coming from different language families to perform translations from the English. Namely, we have studied translations from English to two different Indo-European languages (French (fr) and Spanish (es)), to Arabic (ar) and to two Asiatic languages (Japanese (ja) and Korean (ko)).

We have used the quadratic loss defined in equation (1) and the same set of hyperparameters for all the languages. The results are represented in Figure 2, where we have compared the mean cosine

similarity during training with 200 epochs for each language. As expected, languages belonging to the same family tend to exhibit similar metrics (the graphs corresponding to French and Spanish are very close and the best results are obtained for the two Asiatic languages). It's important to remark the fast convergence of this method, getting to the minimum of the loss usually after less than 50 iterations.

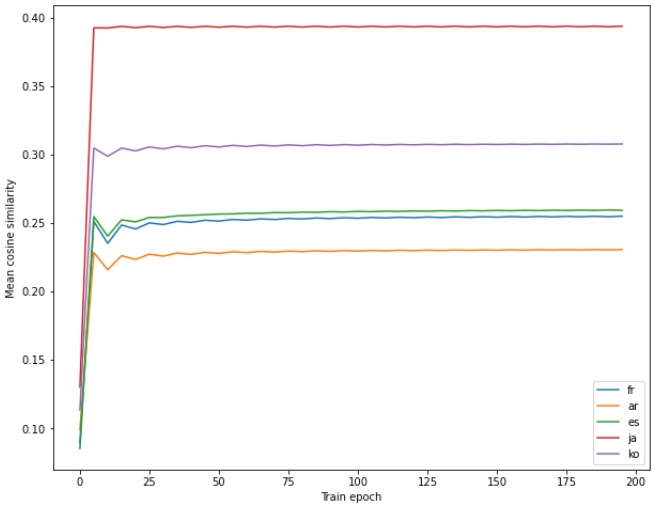

Figure 2: Cosine Similarity Comparison for Translations from English Using Quadratic Loss

If we put together the results obtained for each language for the three methods based on supervised learning, we can also draw important conclusions. As we can see in Figure 3, the method that performs the best for all the languages is the Procrustes method described in Equation (3). Once again, languages belonging to the same linguistic families obtain similar metrics. However, this time the highest accuracy is obtained for the English-Spanish and English-French translations. This result is not surprising, since English belongs to this very same linguistic family and we can expect the similarities between the languages to be reflected in the embedding representations.

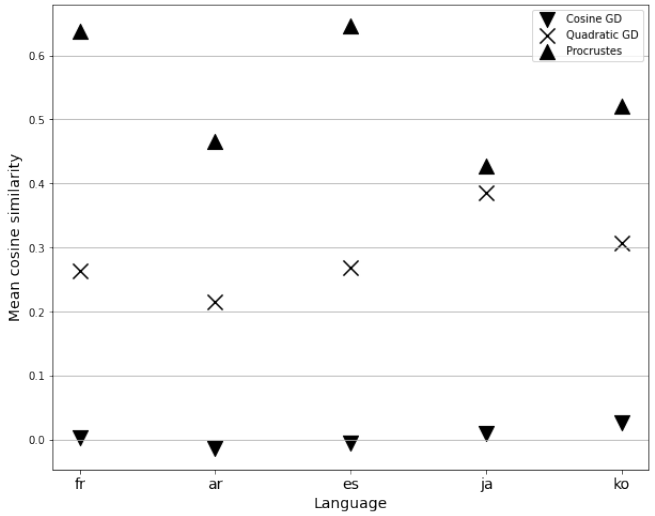

Figure 3: Cosine Similarity Comparison for Different Methods Across Languages.

3.2 Unsupervised Learning

In the previous section we have studied different approaches to machine translation based on preexisting vocabularies with word translations. These methods, as we have seen, use the word pairs in the translation vocabulary to learn a mapping W between the source language space X and the target language space Z. However, there are other methods that don't rely on a preexisting set of translated words. The one that we will present, based on the use of adversarial training, outperforms the previous methods based on supervised learning.

3.2.1 Proposed Method

As with the previous methods, we assume that we already have trained embeddings from the source and the target languages. Let denote this embeddings as $\mathcal{X} = \{x_1, ..., x_n\}$ and $\mathcal{Z} = \{z_1, ..., z_m\}$ for the source and the target languages, respectively, and let W be the linear mapping between them. We implement a model that aims to correctly discriminate between elements of $\mathcal{Z}$ and elements of $W\mathcal{X} = \{Wx_1, ..., Wx_n\}$ generated from $\mathcal{X}$ via the mapping W. We call this first model the discriminator. At the same time we train a model that aims to generate the mapping W in a way that prevents the discriminator from correctly identifying the origin of the mapped elements. This second model is called the generator.

In order to train these two models, we use the same loss function (binary crossentropy) with different target variables for each model. We denote as $P_{\theta_D}(\text{source } = 1|z)$ the probability of a vector Hence, the loss function to optimize for the discriminator can be written as:

$$\mathcal{L}_D(\theta_D|W) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source } = 1|Wx_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\text{source } = 0|y_i) \tag{4}$$

being θ_D the discriminator parameters. In the same way, the loss function for the generator can be written as:

$$\mathcal{L}_W(W|\theta_D) = -\frac{1}{n}\sum_{i=1}^{n} \log P_{\theta_D}(\text{source } = 0|Wx_i) - \frac{1}{m}\sum_{i=1}^{m} \log P_{\theta_D}(\text{source } = 1|y_i) \tag{5}$$

During the training process, we apply Stochastic Gradient Descent to successively minimize these loss functions.

3.2.2 Practical approach

Note : *Due to very demanding computing power, our algorithms aimed at translating only from English to French. As Google Colab servers were continuously crashing and preventing us from training for long periods of time.*

We use the embedding of the French and English dictionaries obtained with fastText. These correspond to monolingual embeddings of dimension 300 trained on Wikipedia corpora; therefore, the mapping W has size 300×300.

Practically, our goal is to train an adversarial network in order to learn the correct mapping matrix W that 'rotates' our source embedding to match the target embedding. Thus, the generator indirectly learns the correct rotation to apply while the discriminator gets better at determining if the output from the generator comes from the true target distribution or fake 'comes from a rotated word embedding'.

In order to do that:

- We define our **generator (mapping)** as one hidden layer network of 300 neurons without bias. Thus, training the weights of the generator is equivalent to learning the matrix W. We initialize the weight to the identity matrix. And we perform the orthogonalization step after each update of W.
- For our **discriminator**, we use a multilayer perceptron with two hidden layers of size 2048, and Leaky-ReLU activation functions. The input to the discriminator is corrupted with dropout noise with a rate of 0.1. We include a label smoothening coefficient of 0.1 in the discriminator's predictions. Also, we make sure to train the discriminator 3 times more than the generator as it is shown in literature to have a better performance.

We use stochastic gradient descent with a batch size of 32, a learning rate of 0.1 and a decay of 0.98 for every 10000 steps both for the discriminator and the generator.

3.2.3 Stopping criteria

Generative Adversarial Networks are notoriously known for being unstable as their performance fluctuates during training. That is because when the generator gets too good, the discriminator cannot differentiate the data, which means his predictions become close to random. So, if we continue training beyond this point, we will back-propagate useless information which will deteriorate our generator (mapping in our case).

There's numerous methods to tackle the instability, but tracking the model during training and choosing afterwards the model that have the best performance is still necessary.

In the classical case, we would have a validation set on which we would run the mapping to know when the W_{best} matrix is reached. However, since we are in a unsupervised case, we can't actually do it as it would mean we have the translation of our dataset.

The paper suggests an unsupervised metric : Cross-Domain Similarity Local Scaling ($CSLS$) that we didn't have enough time to implement but we will nonetheless present in the *Areas of improvement* section.

$\longrightarrow$ Although we are in the unsupervised case, we can reasonably assume that we have access to a very small translated validation set (For instance 3% of our dataset) that will serve us as a stopping criterion.

3.2.4 Results

We remind the reader that we only trained our model for the English-French translation task.

a) Model selection

As stated in the previous section, we used a validation set to select the best performing model. We present the computed accuracy on the validation set throughout training epochs.

We notice that the accuracy of our model increases timidly at the first epochs (1-4) and starts rising until reaching its highest value : 38.24 % before collapsing once again as we explained in the previous section.

b) Evaluation of chosen model :

We run two functions to evaluate our model :

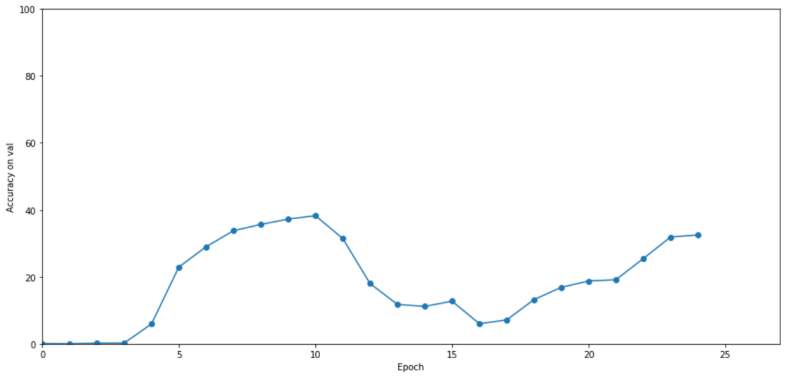

Figure 4: Learning curve - Model selection

- Accuracy : calculates the percentage of words correctly translated using the cosine similarity as a distance in the 1-Nearest-neighbor algorithm
- Mean cosine similarity : calculates how much alignment we have between our rotated test source embedding and the translated target embedding.

Clusters	accuracy (K=1)	mean cosine similarity
Unsupervised GAN	34.36 %	0.47
Procrustes	**69.56** %	**0.63**

Figure 5: Evaluation and comparison with best supervised method

Commentary

- The GAN gave us somewhat good results with about a third of the words correctly translated. We note that this accuracy reaches about 60 % if the true translation is allowed to be among the K=10 neighbors. The mean cosine similarity is 0.47 which is fairly good.
- In comparison with the Procrustean optimization, the unsupervised method have lower performance but it is acceptable given the fact there is no cross-lingual supervision.

c) Revisiting accuracy

The value of accuracy used above results from choosing 1-Nearest-Neighbor, so we explored the accuracy values for $K = 1, .., 10$.

As expected, being too strict with the parameter K (K=1) gave us somewhat of a poor accuracy value, however, relaxing the value of K shows that the algorithm has reasonably good results.

$\longrightarrow$ For instance for K = 5, we have an accuracy of 57.25 %, which means that in 54.1 % of cases, the true translation of the word is at most the fifth furthest neighbor of our rotated word.

3.2.5 Areas of improvement

a) Cross-Domain Similarity Local Scaling (CSLS)

As we have previously discussed in sections 3.2.3, for future work, we could implement the distance Cross-Domain Similarity Local Scaling (CSLS) which was used in the paper to define similarity between mapped source words and target words as :

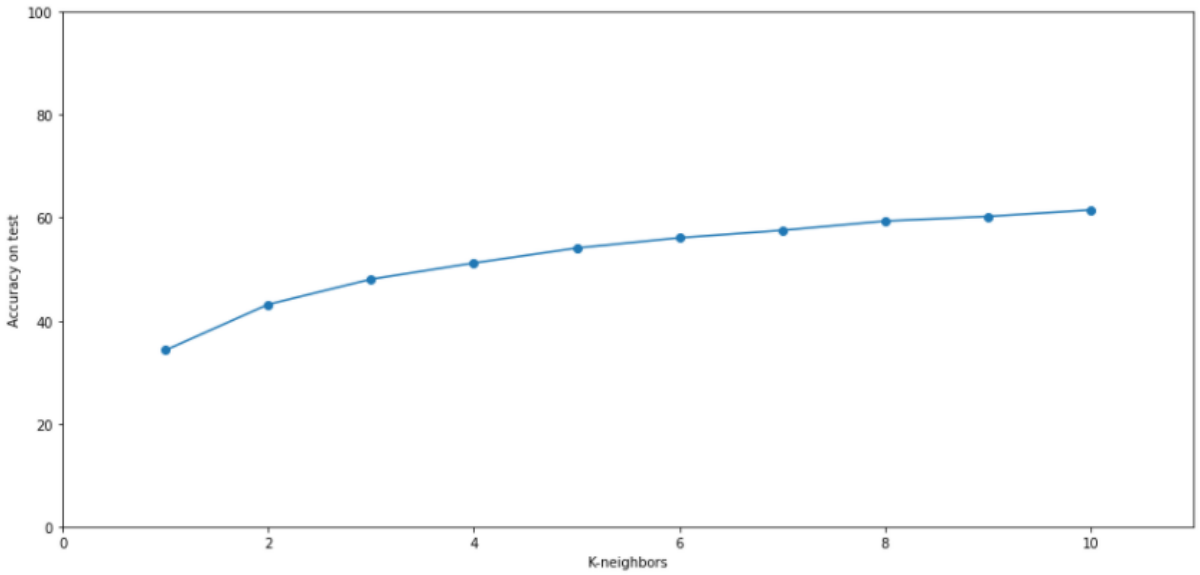

Figure 6: Accuracy on test set for different parameters value of K-NN

$$CSLS(W_{x_s}, y_t) = 2cos(W_{x_s}, y_t) - r_T(W_{x_s}) - r_S(y_t) \tag{6}$$

With :

$$r_T(W_{x_s}) = \frac{1}{K} \sum_{y \in Neighborhood(W_{x_s})}^{K} cos(W_{x_s}, y_t) \tag{7}$$

This process increases the similarity associated with isolated word vectors, but decreases the similarity of vectors lying in dense areas. It was shown in the experiments of the paper to give higher accuracy values.

b) Refinement Procedure

The results of the unsupervised method are usually not on par with the supervised methods such as Procrustes. However, since they work quite well on embeddings of words that are frequent in the training corpus of fastText. One could argue that we could use this unsupervised method as a first step to form a preliminary translation dictionary for the most frequent words.

Then, we would use these translations as anchors for supervised aligning method such as Procrustes. Studying this refinement procedure would be interesting to us.

4 Conclusions

Studying both supervised and unsupervised methods are both hot topics in word translation. Both methods rely on aligning two feature spaces. Supervised methods work very well on this problematic specially the Procrustean optimization method that gives the best accuracy. We notice greater performance for languages that have intrinsic similarities (Engligh/French) in comparison with languages that have different linguistic roots (English/Japanese-Arabic).

The unsupervised way also gave us reasonable results although not on par with its supervised counterpart. However, it is understandable knowing that there is no cross-lingual supervision. Also, we remain optimistic about the adversarial training's efficiency on this problematic especially that we haven't fully exploited some of the methods to refine our training, mainly the methods pointed out in section *3.2.5 Areas of improvement*.

References

[1] C. Xing, D. Wang, C. Liu, and Y. Lin. *Sequence to Sequence Learning with Neural Networks*. Proceedings of NAACL, 2015.

[2] I. Sutskever and O. Vinyals and Quoc V. Le. *Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation*. Advances in Neural Information Processing Systems, 2014.

[3] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, H. Jégou. *Word Translation Without Parallel Data*. ICLR, 2018.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative Adversarial Nets*. Advances in Neural Information Processing Systems, p. 2672–2680, 2014.

[5] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov. *Enriching Word Vectors with Subword Information*. 2016.

[6] Facebook's AI Research (FAIR) Lab. *fastText Resources*. https://fasttext.cc/docs/en/crawl-vectors.html.