Data Science Project - Word Embedding

Laurène Bouksila, Charlotte Durand, Aymeric Jan

November 2020

1 Introduction

In the context of NLP (Natural Language Processing), we call the vector representation of words : Word Embedding. This technique maps words into arrays, thus creating a high-dimensional vector space of words. Monolingual embedding encapsulates semantic relationships between word arguments and this information can later on support bilingual translation through the learning of a transformation matrix that we will call W. This matrix allows the matching of two monolingual corpus and thus, the alignment of word pairs, based on closest similarity criteria through their embedding.

1.1 Word2vec

Abstract

The Word2vec algorithm was created by Tomas Mikolov in 2013, in order to give a possible solution to Word Embedding. This unsupervised learning algorithm produces a high-dimensional vector space representation of words through the use of neural networks.

Note that simple vector algebra (distance, direction, norm) enable us to perceive both
- syntactic matches such as singular/plural, present/past, adjectives/superlatives relations.
As an example, see Fig.2, singular/plural have same direction and distance.
- semantic matches (eg. countries/capitals, family...). See Fig.3 :the representation of king-man+women which corresponds to queen, has the same relation (distance-wise, direction-wise) to king as women has to man.

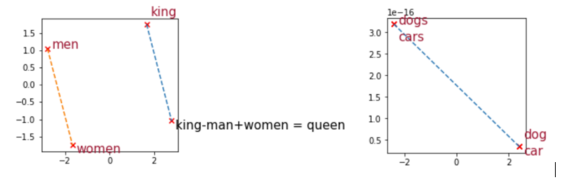

Figure 1: (a) Syntactic visualization relation :singular/plural,
(b) Semantic visualization relation: 'king' - 'man' + 'women' = 'queen'

Architecture

The word2vec approach focuses on displaying multiple degrees of similarity both along syntactic and semantic dimensions through a two-layer neural networks which goal is to reduce the loss function by adjusting its weights.
As you can see on the figure, the network takes as its input a large vector (one-hot vector, filled with zeros except at the index

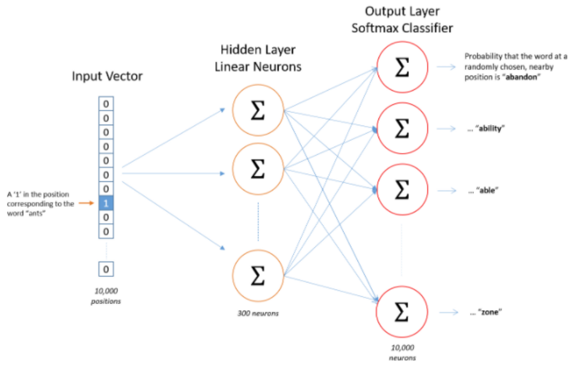

Figure 2: Word2Vec neural network architecture

that represents the word to be embedded). It is passed through the hidden layer whose weights are the word embedding and the output layer outputs the probabilities for this word in the whole vocabulary.
Eventually, at the end of the algorithm, we will have learnt the hidden layer weight matrix (embedding of my vocabulary words).

1.2 Translation Algorithm

By observing corpus of words, we can observe a surprising fact which is that they have the same shape! This means that simple linear transformations (rotation, scaling) might enable us to go from one monolingual corpus to another !
By doing so, we should be able to easily obtain translations between languages.

1.2.1 Theory

We based our approach on a simple fact: one can learn a linear transformation matrix from a bilingual dictionary to overlay two monolingual corpus and translate one word in a language to its corresponding translation in another language.

Unconstrained translation matrix

Indeed, let x_i be the word vector of the i-th word in the source language (monolingual dictionary) and z_i the word vector in the target language, we would like to find a projection matrix W that satisfies the following relation:

$$W = \underset{W \in \mathbb{R}^{p \times p}}{\operatorname{argmin}} \sum_i ||Wx_i - z_i||^2 \tag{1}$$

We first choose the L_2 norm and solve this problem with gradient descent. We define the loss function $f_i : W \to ||Wx_i - z_i||_2^2$ whose gradient is given by $\nabla f_i : W \to 2(Wx_i - z_i)x_i^T$.

Algorithm 1 Calculate W without constraints

```
Require: $X = (x_1, \dots, x_n)^T, Y = (y_1, \dots, y_n)^T, \alpha \in \mathbb{R}$
  Initialize $W$ with random values
  for i $\leq$ n do
    $W \leftarrow W - \alpha 2(Wx_i - y_i)x_i^T$
  end for
  return $W$
```

Orthogonal translation matrix (Constrained version)

An alternative method could be to use the cosine similarity (2) as a distance between Wx_i and z_i.

$$\cos(Wx_i, z_i) = \frac{\langle Wx_i, z_i \rangle}{||Wx_i|| \cdot ||z_i||} \tag{2}$$

To find the matrix W, we first perform gradient descent with the following update rule

$$W_{k+1} = W_k + \alpha \Delta w_k \qquad \text{where } \Delta w_k = x_k z_k^T \tag{3}$$

Then, we make sure that W is orthogonal by changing it's singular values to 1. Since Δw does not depend on W, if α is large enough (i.e. the coefficients of $\alpha\Delta w$ are larger than the ones of W_0), the final value of W will not depends on the learning rate. If $SVD(W) = U\Sigma V^T$, we take $\tilde{W} = UV^T$

The procedure to obtain $\tilde{W}$ is summarize in the following algorithm

Algorithm 2 Calculate $\tilde{W}$ orthogonal

```
Require: $X = (x_1, \dots, x_n)^T, Y = (y_1, \dots, y_n)^T, \alpha \in \mathbb{R}$
  Initialize $W$ with random values
  for i $\leq$ n do
    $W \leftarrow W + \alpha x_i y_i^T$
  end for
  Perform SVD on $W$ such that $W = U\Sigma V^T$
  $\tilde{W} \leftarrow UV^T$
  return $\tilde{W}$
```

1.2.2 Experimental tests: Constrained vs Unconstrained translation matrix implementation

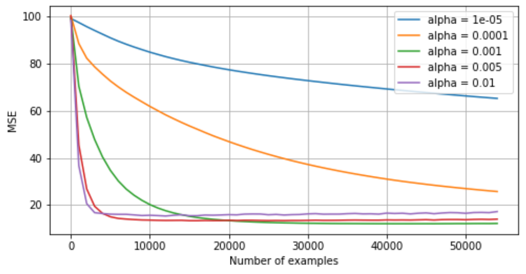

Figure 3: Evolution of MSE during the train of W unconstrained

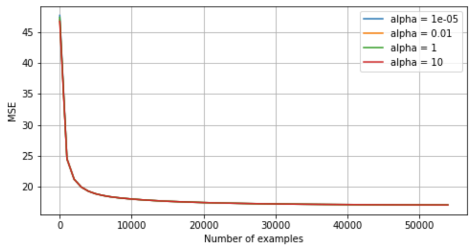

Figure 4: Evolution of MSE during the train of W normalized /constrained

For the *unconstrained* case, we find that $\alpha = 10^{-3}$ seems to be a good choice.Whereas for the case in which the translation matrix is *normalized*, it appears that as long as the learning rate is high enough, we get good accuracy (α >10).

To verify the efficiency of the constrained method, we compare the results obtained with both methods in terms of MSE and accuracy on 500 words.

word in french	est	dans	avec	cette	naissance	décès
unconstrained	which	the	with	however	born	buried
constrained	was	in	with	this	birth	death

Figure 5: Examples of translation with both methods

	MSE	Accuracy
unconstrained	10.63	45.0
constrained	13.97	55.6

Figure 6: MSE and Accuracy between both methods

Even if the MSE seems to be lower with the unconstrained method, forcing W

to be orthogonal slightly improves the accuracy of the translation. Moreover, we can note that the MSE evaluates how close the vectors are, not the words themselves and therefore is not totally representative for performance evaluation as opposite to accuracy.

That is the reason why, for the rest of the project we will use the **constrained implementation** version of the translation matrix W.

2 Supervised Learning - French to English

In this part, we are going to evaluate our algorithm on the translation from French to English. For the following results, W will always be constraint.

First Results and Visual Representation

To check the efficiency of our algorithm, we have two tools :

- the accuracy of our translation on training and testing set : We split N words with a validation$_s$plit to create a training set which will be used to calculate W and to apply translation on this set. The words in the testing set will not be used to calculate W, which give us a proof of the efficiency of our algorithm
- a graphical representation of our translation in the vector space : with a PCA (Principal Component Analysis), we project our 300-dimension space in a 2 dimension space to visualize the translation

On a 3000 Words dataset with a 90% validation split, we obtain a 73% accuracy on the training test and a 80% accuracy on the testing set. To check if we are actually creating the correct W, we use the PCA on a small number of words, before applying W and after applying W.

As we can see on Fig. 7, with the application of W, the same words in different languages are closer in the embedding space.
We are going to see in the next parts the influence of different parameters on our accuracy.

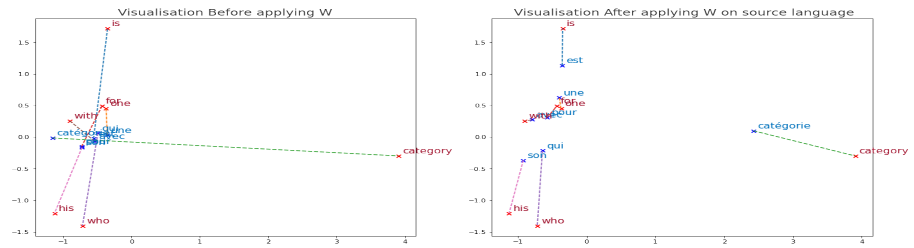

Figure 7: PCA on the target data space and transformation on the key data space before and after applying W

Influence of the initialization of W

With the algorithm using W constraint, we changed the value of α, the learning rate to see the influence of the learning rate and the initialization of W. Those initializations were:

-Matrix of zeros
-Matrix with coefficients following an uniform distribution between -0.2 and 0.2
-Matrix with coefficients following a gaussian distribution centered on 0 with a 0.1 standard deviation
As we can see on Fig. 8 we always obtain the same accuracy for the initialization with 0 : which means the learning rate has no influence on the calculation of W, since we obtain a good accuracy. For the two other distributions, we obtain the same behaviour : for small learning rate, the weight of ΔW multiplied by α is too small in regard to the value of the initialized coefficients. When it is big enough, we get a good accuracy.

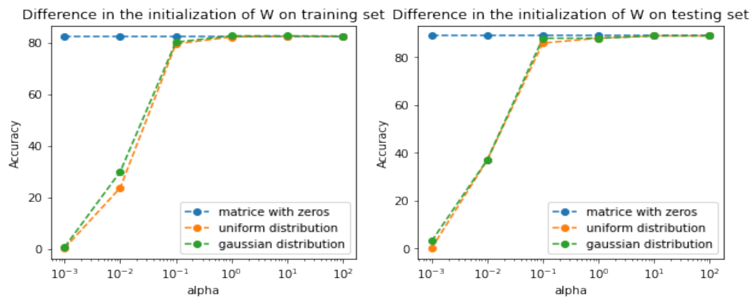

Figure 8: Results on accuracy for different initialization for different value of alpha

Accuracy Evaluation

In this part, we are going to see the influence of different parameters : normalisation of words and distance used for finding the closest word.

All the accuracy are calculated for 1000 words and a 0.9 split between training and testing.

Normalisation At the beginning, we wondered if the normalization of the data in the embedding space has an influence on the accuracy. Thus we created a condition in the program which allow the user to normalize or not along the axis of dimension space (each word has a norm of one). We get the following results on accuracy calculated with cosine distance.

	With Normalisation	Without Normalisation
Training Set	82.33	82.44
Testing Set	90,0	90,0

Figure 9: Accuracy for normalized and unnormalized data

	Cosine Distance	Euclidian Norm
Training Set	82.33	76.33
Testing Set	90,0	79,0

Figure 10: Accuracy for different distances

As we can observe, there is no difference in the accuracy whether we normalize or not the data. Of course,we checked that the data were not normalized at the beginning. The explanation isn't really clear for us.

Distance Calculation When we find the closest word in the target langage, we need to use a distance. Two distances were evaluated for this : the cosine distance defined above and the classical Euclidian Norm. Let's see the influence of the choice of the distance on our algorithm :
The euclidian results are less interesting than the cosine distance. This was actually expected since the cosine distance measure by definition the similarity between two vectors.

3 Different Languages, alphabets Evaluation

3.1 Visual Representation

In this part of the project, we have selected the constrained version for the translation matrix implementation with the Gaussian initialization (0-1).We will work on different languages: English, Spanish and Hebrew. The motivation for these choices reside in the fact that we want to evaluate the influence of the alphabet type and the performances over various different languages translations.

Hereafter, we can observe the visual representation of Spanish obtained through Principal Component Analysis as before.

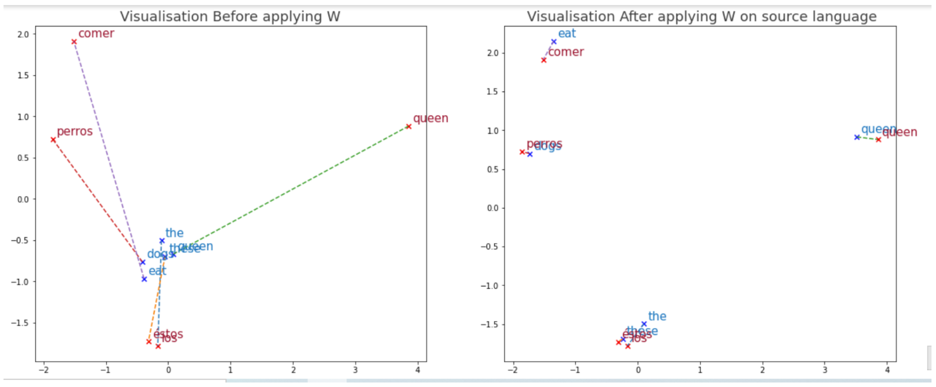

Figure 11: PCA English-Spanish - 5 words

3.2 Alphabet Influence on accuracy

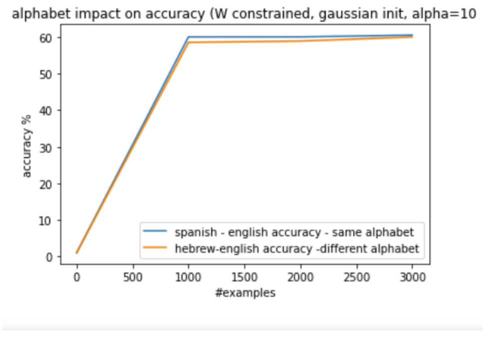

Figure 12: Alphabet impact on accuracy

In this section we will focus our interest on the way the accuracy is affected when dealing with different kind of alphabets.We run this experiment on 3000 words, with W (translation matrix) constrained, α= 10 (learning rate) and cosine distance as evaluation metric.
We expect a higher accuracy for languages with the same alphabet (English-Spanish) rather than on different ones (English-Hebrew).

As we can see, there exists a slight difference in the accuracy between translations Spanish-English and Hebrew-English. This tendency gets negligible as we are dealing with more and more words, but this can be explained by the fact that we are translating languages with different alphabets in the second case.

References

[1] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *CoRR*, abs/1710.04087, 2017.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119. Curran Associates, Inc., 2013.

[3] Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado, May–June 2015. Association for Computational Linguistics.