The Journal of Supercomputing (2021) 77:12320–12335
https://doi.org/10.1007/s11227-021-03743-2

Considerations about learning Word2Vec

Giovanni Di Gennaro[1] · **Amedeo Buonanno**[2] · **Francesco A. N. Palmieri**[1]

Accepted: 13 March 2021 / Published online: 6 April 2021

Abstract
Despite the large diffusion and use of embedding generated through Word2Vec, there are still many open questions about the reasons for its results and about its real capabilities. In particular, to our knowledge, no author seems to have analysed in detail how learning may be affected by the various choices of hyperparameters. In this work, we try to shed some light on various issues focusing on a typical dataset. It is shown that the learning rate prevents the exact mapping of the co-occurrence matrix, that Word2Vec is unable to learn syntactic relationships, and that it does not suffer from the problem of overfitting. Furthermore, through the creation of an ad-hoc network, it is also shown how it is possible to improve Word2Vec directly on the analogies, obtaining very high accuracy without damaging the pre-existing embedding. This analogy-enhanced Word2Vec may be convenient in various NLP scenarios, but it is used here as an optimal starting point to evaluate the limits of Word2Vec.

Keywords Word embedding · Natural language processing · Neural networks

1 Introduction

In *Natural Language Processing* (NLP) problems approached with *neural networks*, individual words, that typically belong to large vocabularies, must be transformed into compressed representations. Although the state-of-the-art of NLP is today

✉ Giovanni Di Gennaro
giovanni.digennaro@unicampania.it

Amedeo Buonanno
amedeo.buonanno@enea.it

Francesco A. N. Palmieri
francesco.palmieri@unicampania.it

[1] Dipartimento di Ingegneria, Università della Campania "Luigi Vanvitelli", Via Roma, 29, 81031 Aversa, CE, Italy

[2] ENEA, Department of Energy Technologies and Renewable Energy Sources, Research Centre of Portici, P. E. Fermi, 1, Portici, NA, Italy

almost totally based on the use of Transformers [10, 30, 34], the difficulty of training such structures (both related to computational costs and the need for huge datasets) often leads to a preference for different approaches [5, 11, 17, 18, 26] where each word needs to be individually coded.

In these cases, it is therefore natural to look for codings that account for semantic relationships between words (what in [33] is called *attributional similarity*). This leads to the creation of a so-called *word embedding* (sometimes named "semantic vector space" or simply "word space"), i.e., a continuous vector space in which the relationships among the vectors is somehow related to the semantic similarity of the words they represent. The ways of creating these spaces are almost entirely based on the *distributional hypothesis* [14–16, 25], that is, on the idea that contextual information alone is able to define the semantic connections that exist between individual words.[1] Through the use of very large corpora, these models typically produce vector spaces with hundreds of dimensions to grasp different levels of similarity between words. Similarity proportions such as "Man is to Woman as King is to Queen" are thus reproducible through vector arithmetic [24], allowing to express the relationship between words as geometric proximity. For example, the sum vector obtained from the equation "King" − "Man" + "Woman" returns the vector relative to "Queen" as the closest neighbor, which is obviously extremely useful in NLP. It should be noted that, in general, the uniqueness of the vectors is not mathematically guaranteed but is always supposed to be verified, given the very low probability of the opposite happening.

Starting from the work in [9], such semantic vector spaces began to be learned through neural models. To date there are numerous word embedding models (a fairly complete list is present in [2]), but the main scheme that makes use of neural networks is known by the name of **Word2Vec** (W2V) [22, 23]. The production of a word embedding through W2V can take place in two different ways: *Continuous Bag-of-Words* (CBOW) and *Skip-Gram* (SG). The two approaches rely on different management of the input and the output variables, but basically use the same structure of the network. In the following, we will focus only on the SG approach, which is the most used in practice and studied in the literature [3, 19, 21]. The success of this structure is undoubtedly linked to its performance which on the task of analogies proves better than both classic techniques, such as *Singular Value Decomposition* (SVD) [19, 20] and *Latent Semantic Analysis* (LSA) [3, 4], and modern count-based methods, such as GloVe [20, 27, 29].

Although many authors have tested Word2Vec on analogies [13, 19–21, 24, 28], rarely enough attention has been given to the modalities in which such embeddings are obtained. In this work, we try to shed light on the performance of W2V as the number of epochs changes, showing how the particular behavior of the learning rate justifies an individual analysis of the single epochs. This innovative way of proceeding highlights elements of extreme interest, including: the inability of W2V to learn syntactic, the absence of overfitting and the stabilization of learning around

[1] For this reason, in computational linguistics it is preferred to use the term "distributional semantic models" [6, 7], or generically "distributed representation".

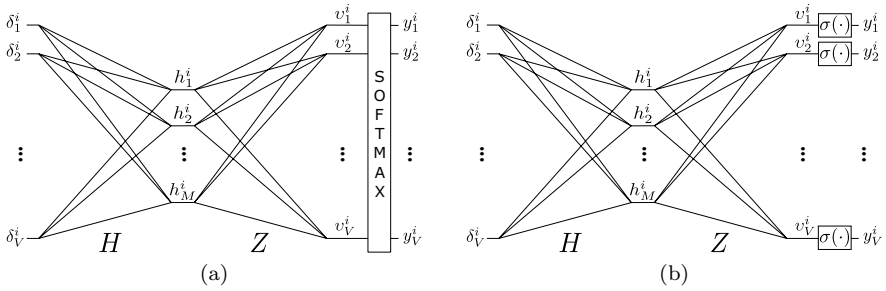


Fig. 1 SG architecture with two activation functions: **a** softmax, **b** sigmoid

a maximum value. Finally, it is shown how to improve W2V through an ad-hoc training directly on the analogies, achieving high accuracy by introducing very few adjustments to a pre-trained embedding. This process highlights the limitations of Word2Vec, demonstrating that it is insensitive to better starting conditions.

In Sect. 2, the details of W2V are introduced, in Sect. 3, the elements that emerge from the tests performed are examined, in Sect. 4, the analogy-enhanced version of W2V is shown. Conclusions and comments are included in Sect. 5.

2 Word2Vec

Given a vocabulary $\mathcal{V} = \{w^1, w^2, \dots, w^V\}$, the W2V SG structure (Fig. 1) derives from a two-layer neural network with linear activation (identity) in the hidden layer and no bias, mathematically expressed[2] as:

$$\boldsymbol{h}^i = \boldsymbol{\delta}^i H, \quad \boldsymbol{v}^i = \boldsymbol{\delta}^i HZ, \quad \mathbf{y}^i = \varphi(\boldsymbol{\delta}^i HZ) \tag{1}$$

where $H \in \mathbb{R}^{V\times M}$, $Z \in \mathbb{R}^{M\times V}$, $\boldsymbol{\delta}^i$ is the V-dimensional one-hot row vector relative to the generic word w^i at the input of the neural network, $\boldsymbol{h}^i \in \mathbb{R}^M$ is its related embedding, $\boldsymbol{v}^i \in \mathbb{R}^V$ is the linear combination before the activation functions, and $\mathbf{y}^i \in \mathbb{R}^V$ is the network output after the activation function $\varphi(\cdot)$. The dimensions of the input and output layer of this network are therefore the same and equal to the size of the vocabulary $V = |\mathcal{V}|$, while the size M of the hidden layer represents a hyperparameter chosen arbitrarily to be much smaller than V. Figure 1 shows the architecture with two different activation functions that will be discussed later.

2.1 The vocabulary

W2V training starts from a natural language text corpus $\mathcal{C}$ consisting of a sequence of $|\mathcal{C}|$ words $(w[k])_{k\in\mathcal{K}}$, with $\mathcal{K} = \{1, 2, \dots, |\mathcal{C}|\}$, where each distinct word

[2] We will follow the classic convention that uses bold to represent vectors, so that their components are more easily distinguishable.

w appears throughout $\mathcal{C}$ a number of times equal to $\mu(w)$. The original corpus $\mathcal{C}$ is pre-processed to produce a smaller *reference corpus* $\widetilde{\mathcal{C}}$, from which all the words that occur less than T times are eliminated:

$$w \in \widetilde{\mathcal{C}} \iff \mu(w) \geq T \tag{2}$$

This pre-processing removes writing errors, or words that are too rare to be considered in the embedding. Then the distinct words that belong to the reference corpus $\widetilde{\mathcal{C}}$ constitute the vocabulary $\mathcal{V}$, for which the empirical probability is:

$$P(w^j) = \frac{\mu(w^j)}{\sum_{l=1}^{V} \mu(w^l)}, \quad j = 1, \ldots, V. \tag{3}$$

2.2 Learning the embedding

According to a criterion that will be specified in the following, the training of the network requires a set of input/output (i, o) ordered pairs $\mathcal{P} = \{(w^i[\kappa], w^o[\kappa])\}$, generated in advance from the reference corpus $\widetilde{\mathcal{C}}$. Every single pair of words (w^i, w^o) is associated with its relative one-hot vectors $(\boldsymbol{\delta}^i, \boldsymbol{\delta}^o)$ that represent, respectively, the input and desired output of the network. Training takes place through a classic *stochastic gradient descent* (SGD) algorithm with instantaneous *categorical cross-entropy loss* and gradient:

$$\begin{aligned} \mathfrak{L}^{io} &= -\sum_{j=1}^{V} \delta_j^o \ln y_j^i \\ &= -\ln\left(\frac{\exp(v_o^i)}{\sum_{j=1}^{V}\exp(v_j^i)}\right) \end{aligned}; \quad \begin{cases} \dfrac{\partial \mathfrak{L}^{io}}{\partial v_o^i} &= \left(\dfrac{\exp(v_o^i)}{\sum_{j=1}^{V}\exp(v_j^i)} - 1\right) \\ \left.\dfrac{\partial \mathfrak{L}^{io}}{\partial v_n^i}\right|_{n\neq o} &= \dfrac{\exp(v_n^i)}{\sum_{j=1}^{V}\exp(v_j^i)} \end{cases} \tag{4}$$

where δ_j^o, y_j^i and v_j^i represent the j-th element of the vectors $\boldsymbol{\delta}^o$, $\boldsymbol{y}^i$ and $\boldsymbol{v}^i$, respectively, and when at the network output there is a softmax activation function (Fig. 1a). Note that v_o^i denotes the component of $\boldsymbol{v}^i$ corresponding to the non null element of $\boldsymbol{\delta}^o$.

Since the use of pure softmax at the output layer would represent an excessive computational cost (as the network, although simple, has a decidedly large number of parameters due principally to the dimension of the vocabulary V), the typical alternatives fall either on adopting an approximation of it (called "hierarchical softmax", and which we will not discuss here), or resorting to a technique known as "negative sampling" [23]. In this case, the network is modified to the architecture of Fig. 1b, which has sigmoid activation function on each neuron of the output layer. The computational cost is reduced by backpropagating only N randomly chosen errors of the $V - 1$ ones, relating to the output words w^n that do not correspond with the word w^o present in the single pair (i.e., $n \neq o$). The negative sampling then turns the problem into a multi-label classification one, where the instantaneous *binary cross-entropy loss* and its gradient are:

$$\begin{aligned}\mathcal{L}^{io} &= -\ln y^i_o - \sum_{j=1}^{N} \ln(1-y^i_{n_j})\Big|_{\substack{n_j \neq o \\ n_j \sim P_{\text{neg}}(w^{n_j})}} \\ &= -\ln\left(\frac{1}{1+\exp(-v^i_o)}\right) \\ &\quad -\sum_{j=1}^{N} \ln\left(\frac{1}{1+\exp(v^i_{n_j})}\right)\Bigg|_{\substack{n_j \neq o \\ n_j \sim P_{\text{neg}}(w^{n_j})}}\end{aligned} \quad ; \quad \begin{cases} \frac{\partial \mathcal{L}^{io}}{\partial v^i_o} = \frac{-1}{1+\exp(v^i_o)} \\ \frac{\partial \mathcal{L}^{io}}{\partial v^i_{n_j}} = \frac{1}{1+\exp\left(-v^i_{n_j}\right)} \end{cases} \tag{5}$$

The N random words of Eq. (5), that act as "negative" set for that single training pair, are sampled from the heuristically modified "unigram distribution" of the words in the corpus $\widetilde{\mathcal{C}}$ [24]:

$$P_{\text{neg}}(w^n) = \frac{\mu(w^n)^{3/4}}{\sum_{j=1}^{V} \mu(w^j)^{3/4}}. \tag{6}$$

2.3 Pairs generation

Since the presence of common words (such as "the", "of", etc.) is very high in regular texts, a classic problem in creating a set of training pairs lies in making sure that they are not considered too often [23]. To achieve this, W2V modifies the true word empirical probability by defining a "keeping probability" as:

$$P_{\text{keep}}(w^n) = \left(\sqrt{\frac{P(w^n)}{\zeta}} + 1\right)\frac{\zeta}{P(w^n)} \tag{7}$$

where ζ is a heuristically-determined value, typically set between 10^{-3} and 10^{-5} (in the following we take it equal to 10^{-5}). The nonlinear transformation (7) is highly peaked around small probability values and reduces the effect of very frequent words. Each single word w of the corpus is then analysed using the following procedure: take a uniformly distributed random values $r \sim \mathcal{U}_{(0,1)}$, i.e., extracted according to a uniform distribution in [0, 1], if $r < P_{\text{keep}}(w)$ the word becomes a "central word", otherwise is discarded.

The corpus $\widetilde{\mathcal{C}}$ is also divided into sentences (each containing at most a maximum number of words). Once a central word has been chosen, two windows of words are built within the sentence: one towards its right and the other one towards its left. The words that belong to these windows constitute the "context words" for that central word. The window size is not fixed but varies dynamically and randomly on each epoch and for each central word considered, according to a uniform distribution in [1, W] (with W hyperparameter defined at the beginning) [22]. In this way, the words closer to the central words are considered more times but also words further away are too. Also note that, being limited by the extremes of the sentence, the two windows do not always have the same size. Finally, each central word is associated with

each of the words in its context to generate the training pairs. For each pair, the central word represents the input while the context word the output.

2.4 Word embedding evaluation

The main problem after having obtained a word embedding is precisely how to test it. Unfortunately, semantic proximity is indeed difficult to prove, and probably all tests (whether *extrinsic* and *intrinsic* [32]) prove arbitrary or subjective in evaluating this property. The use of analogies, however, has been a standard approach for some time [13, 19–21, 24, 28], although it should be noted that they are certainly not perfect. For example, if we consider the semantic proportion "Athens is to Greece as Tehran is to..." (and although the correct answer is undoubtedly "Iran") it is hard to assess whether or not the possible answer "Persia" should be declared as an error. Natural language is in fact usually highly polarized, as it also depends on socio-cultural influences. However, the use of triads of words certainly makes the search field of the desired more limited than all other possible tests, making it one of the most important tests in this field.

In the present study, we use the most common test set of analogies, known as *Google Set* and included in the original distribution of the W2V package [22].

It consists of 19,544 analogies divided into 14 categories, typically grouped into "semantic" (5 categories with 8869 analogies) and "syntactic" (9 categories with 10,675 analogies) macro-areas; an example table is presented in [22]. Each of the analogies in this dataset can be written symbolically as:

$$w^a : w^{a^\star} = w^b : w^{b^\star} \tag{8}$$

where typically the word $w^{b^\star}$ is chosen as the test target. For example, if we have: "Man: Woman = King: Queen", with w^a (Man), $w^{a^\star}$ (Woman) and w^b (King), we expect to fill-in the answer with $w^{b^\star}$ (Queen). In all the tests performed, however, it was decided to totally neglect all the analogies that contain one or more words not present in the vocabulary.[3] Nevertheless it is good to specify that, since the goal of this work is not to compare different models, this choice is completely irrelevant from our point of view.

Following previous works [13, 21, 24], to provide the answer for the single analogy we use the "classical" cosine distance, also known as 3CosAdd [19]. The cosine distance has the advantage of not excessively weighing the amount of contributions obtained from the backpropagation of the gradient during the training phase, which can lead to excess increase or decrease of the single vector norm. In this way, the balance achieved with respect to the other vectors present is mainly considered. More specifically (assuming the following relations: $w^a \rightarrow \boldsymbol{\delta}^a \rightarrow \boldsymbol{h}^a$, $w^{a^\star} \rightarrow \boldsymbol{\delta}^{a^\star} \rightarrow \boldsymbol{h}^{a^\star}$, $w^b \rightarrow \boldsymbol{\delta}^b \rightarrow \boldsymbol{h}^b$), the answer will be the word whose index is:

[3] In practice, the use of *Text8* for model training (as described below) therefore reduces the initial set of analogies to 17,827 (of which 7416 semantic and 10,411 syntactic).

$$\underset{\boldsymbol{h}^j \in \mathcal{H}_e}{\text{argmax}} \cos(\boldsymbol{h}^j, \boldsymbol{h}^s) \qquad \text{with} \quad \begin{cases} \cos(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\boldsymbol{\alpha}\boldsymbol{\beta}^\top}{\|\boldsymbol{\alpha}\|\|\boldsymbol{\beta}\|} \\ \boldsymbol{h}^s \qquad\quad = \frac{\boldsymbol{h}^b}{\|\boldsymbol{h}^b\|} - \frac{\boldsymbol{h}^a}{\|\boldsymbol{h}^a\|} + \frac{\boldsymbol{h}^{a^\star}}{\|\boldsymbol{h}^{a^\star}\|} \end{cases} \tag{9}$$

where the set $\mathcal{H}_e$ is the collection of all the embeddings except $\boldsymbol{h}^a$, $\boldsymbol{h}^{a^\star}$ and $\boldsymbol{h}^b$. In the network of Fig. 1, this corresponds to an amended embedding matrix:

$$H_e = \text{diag}\,(\mathbf{1} - \boldsymbol{\delta}^s)H \tag{10}$$

where $\mathbf{1}$ is the V-dimensional all-ones vector, and $\boldsymbol{\delta}^s = \boldsymbol{\delta}^b + \boldsymbol{\delta}^a + \boldsymbol{\delta}^{a^\star}$. By eliminating the rows relating to the words of the analogy used in the first part, the amended matrix H_e reflects the classic attitude that seeks the solution in the word space from which the words used in the sum have been excluded. Note that this also implicitly imposes that all analogies are constructed so that the searched word is never contained in the triad used in the query.

Matrix H can also be normalized by row in advance, generating a new matrix $\widehat{H}$ that now contains all normalized embedding $\hat{\boldsymbol{h}}$. This preventive normalization allows to calculate the cosine distances through simple scalar products, since (by setting $\widehat{H}_e = \text{diag}\,(\mathbf{1} - \boldsymbol{\delta}^s)\widehat{H}$ and $\hat{\boldsymbol{h}}^s = \hat{\boldsymbol{h}}^b - \hat{\boldsymbol{h}}^a + \hat{\boldsymbol{h}}^{a^\star}$) it is possible to observe that:

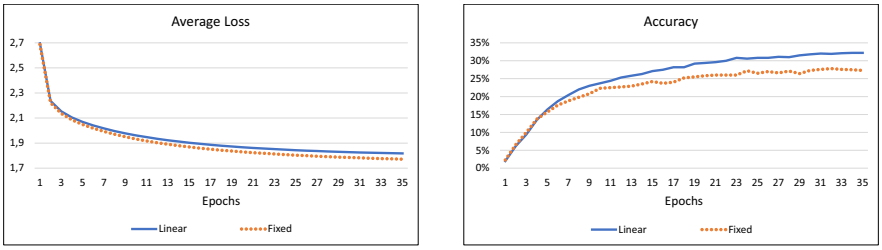
$$\underset{\hat{\boldsymbol{h}}^j \in \widehat{\mathcal{H}}_e}{\text{argmax}} \cos(\hat{\boldsymbol{h}}^j, \hat{\boldsymbol{h}}^s) = \text{argmax}\, \hat{\boldsymbol{h}}^s \widehat{H}_e^\top \tag{11}$$

As typical, if the maximum of $\hat{\boldsymbol{h}}^s \widehat{H}_e^\top$ is in the $b^\star$ index, i.e., in the index position of the word $w^{b^\star}$, the response of the network is considered correct (increasing the accuracy), otherwise it is considered incorrect.

3 The importance of learning time

In this paper, we focus on various issues related to the results obtained from training W2V. In our experience, also in obtaining W2V for the Italian language [12] and in its usage [11], we found that some important choices have become so common that they are used almost mechanically, without questioning about their effectiveness. More specifically, what is the correct number of epochs that need to be used before we can declare an embedding satisfactory? What is the role of the learning rate? More importantly: what is the effect of these choices when performances are studied in comparison for both accuracy and loss?

In fact, regardless of corpus used (which certainly impacts strongly on the quality of the generated embedding), no one seems to have ever bothered to analyse the behavior of the W2V as the number of epochs varies, sometimes making comparisons with other word embedding methods without even reporting this parameter [7, 13, 19–21, 24, 28, 32]. Our goal is therefore precisely to fill this gap, observing the behavior of the embending in the different epochs as the training hyperparameters vary.



(a) (b)

Fig. 2 Comparison between loss and accuracy in relation to the choice of the learning rate

We describe here our experience on several simulations applied to the classic *Text8* corpus, composed of the first 100 MB of cleaned text of the English Wikipedia dump of Mar. 3, 2006. From this corpus, all the words repeated less than $T = 5$ times have been removed, thus obtaining a vocabulary composed of $V = 71{,}290$ words. Although much larger corpora are usually used for more recent W2V embedding [1, 12], we chose this one because we consider it sufficiently typical for focusing on the issues outlined above.[4] On the other hand, the aim of this study is primarily to highlight the relationships that exist between the different results. Since the modification related to the change of the hyperparameters is substantially linked to the training methods, the relationships between them can be rightly considered independent from the corpus (to which only a modification of the absolute accuracy values, which are secondary here, will be linked).

3.1 Learning rate

The first important consideration to make, also to better understand the tests performed later, concerns the learning rate. A typical W2V training using the SGD is in fact based on a variable learning rate, where a starting value (generally in the order of 10^{-2}) and a final value (generally in the order of 10^{-4}) are defined with a step size decaying linearly as a function of the number of epochs used. This classical machine learning technique [8, 31] should aim to decrease the loss, allowing a better approach to the minimum compared to a fixed learning rate.

Figure 2a shows the behavior of the average loss, with a linear and a fixed rate, as the number of epochs progresses. Note that already after a few epochs, and contrary to what one would expect, the fixed rate (here 10^{-2}) finds a better minimum than a typically used degrowth rate (here from 10^{-2} to 10^{-4}). This may be due to the highly non-convex nature of the cost function, which should therefore lead to preferring a different choice from the one commonly used.

[4] This corpus is commonly used for W2V training and is composed of a sufficiently large number of words (17,005,208 words), but above all it is "typical" in the sense that the internal structure (phrases extracted from Wikipedia) is of the same type as most usable corpora.

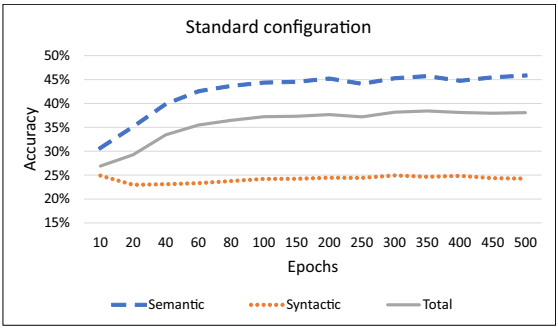


Fig. 3 Tests with $\eta = 0.025 \div 0.0001$, $N = 5$, $W = 5$ and $M = 300$

The surprising result on the analogy test set shown on Fig. 2b is that exactly the opposite happens with respect to the loss function: a substantial increase in the accuracy (from 27.3 to 32.2%) is obtained for the variable learning rate.

Our interpretation of the results is that W2V maximizes accuracy not only by minimizing the loss function (which means mapping the co-occurrence matrix in the best possible way), but also by trying to reduce the link between words and their distribution as the connection between them increases. Probably, the linear decrease of the learning rate allows to fix the rarer words within the embedding space, giving them a more and more reduced possibility of movement; because the second matrix Z is gradually less "conditioned" by these words. In addition to minimizing the loss, the use of many epochs is therefore also necessary to make the learning rate decrease smoother, allowing a gradual stop of the movement of vectors within the embedding space.

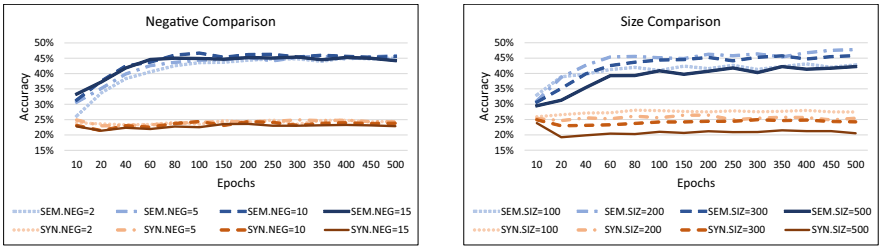
3.2 Simulations and comparisons between hyperparameters

In order to understand what happens when the number of epochs changes, you cannot therefore simply train W2V over a large number of epochs and see how the training proceeds at each step. In fact, in order to have a correct computation of the decreasing learning rate for the current trial, we must ensure that the learning rate reaches its minimum.

Using this different way of looking at learning outcomes over different epochs, extremely interesting behaviors (never been highlighted before, to our knowledge) are observed. Each test presented below, therefore, was performed respecting this rule, and the results were averaged over more simulations.[5] In addition, the tests shown were performed avoiding to parallelize the code within the single epoch,[6] since the SGD would strictly not allow parallelization and an attempt was made to avoid possible influences of uncontrollable elements.

[5] In particular, every single point of every single graph (relative to every single epoch shown) was performed for at least 20 times.

[6] However, the code was parallelized on different processors for different epochs (which are considered independent of each other).



(a) (b)

Fig. 4 Tests by varying a single element of the standard configuration

Following this way, Fig. 3 shows the trend of a W2V training with: learning rate η from 0.025 to 0.0001, negative samples $N = 5$, maximum window $W = 5$ and embedding size $M = 300$. The analogies used for the accuracy test have been divided into the two categories *syntactic* and *semantic* as described in Sect. 2.4. The graph reports the average percentages on the two categories, so that the incidence is assessed regardless of the absolute number of elements present in each category.

From the curves, it can be observed that, relating to the syntactic part, the quality of the embedding is essentially independent from the number of epochs. This element is also present in all the other simulations and highlights an extremely important fact: W2V does not seem to be able to learn syntactic. This means that the various comparisons between W2V and different word embeddings cannot take into consideration datasets mainly based on syntactic, because this would induce a significant bias in the evaluation of the results.

Furthermore, W2V does not really seem to go overfitting, as in fact the trend on the test set does not decrease but stabilizes. This means that there is a "saturation" value for learning W2V which should always be reached in order to perform a correct comparison with any other possible word embedding method.

3.2.1 Negative sampling

The results of other tests for different choices of the parameter for negative sampling ($N = 2, 5, 10, 15$) are shown in Fig. 4a. It can be observed that, except for a few epochs with larger values, as the epochs progress, the various configurations tend to be almost identical (especially beyond 300 epochs). Moreover, the speed of convergence to the steady state value for values $N > 10$ does not seem to undergo variations, allowing this choice to be relaxed (for example for computational reasons).

3.2.2 Size of the embedding space

In Fig. 4b, a comparison with variable size of the embedding space is reported ($M = 100, 200, 300, 500$). The results quite clearly reveal how the quality of embedding is very tied to its size.

The peculiarity, however, is that the achievement of better performances under the semantic profile (reached with dimensions approximately equal to the square

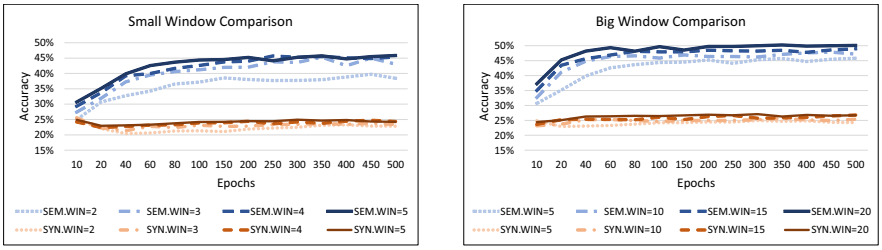


Fig. 5 Tests of a 300-dimensional space when the window changes

root of the vocabulary) does not coincide with the performance of the syntactic part, which is always worse. This shows how the accuracy of the syntactic part is actually determined only by the compression level of the intermediate space, confirming even more how the W2V training is not able to condition it. It should be noted that a larger space makes things worse from every point of view, probably because this makes the network able to better map the matrix of co-occurrences (paradoxically managing to reduce the loss better).

3.2.3 Window size

On the other hand, the change in the window size between small values (from 2 to 5), shown in Fig. 5a, seems to be of little importance. In fact, neglecting window size 2, which in 50% of cases involves a single word to the right and left (showing a clear inability to approach the performance of the other windows), it can be observed that small differences on small windows tends, at increasing epochs, to converge towards similar accuracy.

Different is the case of the results reported in Fig. 5b, and obtained for large window sizes ($W = 5, 10, 15, 20$). Here, in fact, a fixed (and sufficiently large) size increment for the window leads, in steady state conditions, to an equally rigid increase in performance; which are translated upwards both as regards the semantic and the syntactic part (albeit in a reduced way).

Larger windows also tend more rapidly to high accuracy, almost contradicting the distributional hypothesis. In reality, remembering the paragraph Sect. 2.3 and given a window of size m, the probability of forming a pair for a word placed at a distance of d words from the considered one, turns out to be equal to:

$$p = \frac{m - d + 1}{m} \tag{12}$$

and therefore the increase in the size m of the window also increases the probability of the closest words to form a pair with it. It seems to underlie the need for a Gaussian window, which weighs more the neighboring words. Nevertheless, the use of a larger maximum window W also leads to the creation of a larger training dataset, which allows to find a better connection between words.

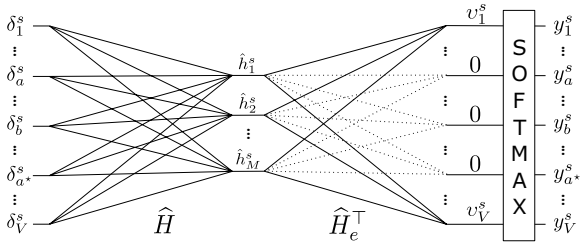


Fig. 6 Network scheme used for training on the analogies

This could also be the reason for the improvement of the accuracy on the syntactic part, which is probably only linked to the relationship between the pairs to be mapped and the space available.

Finally, the "positive" conditioning of a very common distant word will certainly be canceled by the many "negative" conditioning that will occur, while the less common words will create exceptions, fortifying connections even if placed at a greater distance. In fact, it should be noted that a typical W2V training (mainly to avoid high computational costs) does not consider shuffling all the training pairs, but at the most it mixes sentences. In other words, SGD training on the word pairs often takes place in the order in which the words occur within the sentence, and therefore even if a distant word falls within the window it would also be conditioned by the words between them.

4 Analogy-enhanced Word2Vec

In this section, we report the results on training W2V directly on analogies. The structure used (shown in Fig. 6) reflects the test phase through a neural network with linear activation (identity) in the hidden layer, but adding a softmax activation at the output. Note how the connections have been amended. The softmax function tends to focus the backpropagated gradient mainly on the vectors "closest" to the vector of interest, while modifying the other vectors (which however produce some relevance) as little as possible.

The loss is calculated through the cross-entropy function $\mathfrak{L}$ (Eq. 4) and assuming that the desired output is the one-hot vector $\boldsymbol{\delta}^{b^\star}$ relative to the fourth word $w^{b^\star}$:

$$\widehat{H} = \widehat{H} - \eta \frac{\partial \mathfrak{L}}{\partial \widehat{H}} = \widehat{H} + \eta \left(\boldsymbol{\delta}^{b^\star} - \frac{\exp(\boldsymbol{v}^s)}{\sum_{j=1}^{V} \exp(v_j^s)} \right)^\top \hat{\boldsymbol{h}}^s \tag{13}$$

Due to the relatively few analogies present, training of the W2V cannot be based solely on them. Therefore, the starting matrix is taken from an already trained network on 40 epochs, with standard configuration parameters.

Further training on analogies was performed for only 15 epochs, using a subset of 20% of them with a fixed learning rate η equal to 0.01 and normalizing all the vectors at the beginning of each training step (i.e., at each matrix

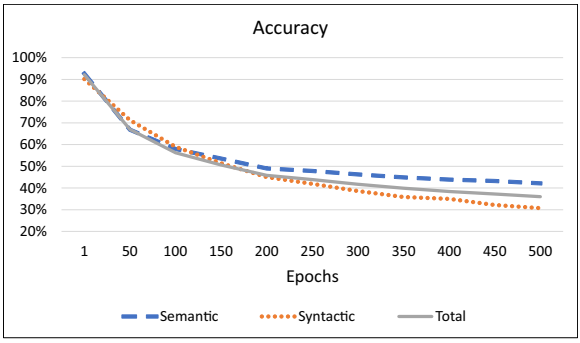


Fig. 7 Accuracy trend following an improvement in embedding

modification). Although the analogies are randomly permuted before being chosen, even such a simple configuration leads to results around 97% of accuracy on the whole set. This result is however conditioned by the structure of the dataset, which always uses the same words and permutes them within the various analogies. Despite this, it is important to note that at the end of this training process only about 450 vectors relating to the searched words undergo an angle shift, while all the other vectors, remain practically immobile. In other words, the network better positions only the vectors that do not provide the correct solution, and the fact of having amended the output matrix allows this shift to be made by fixing the other three points in space.

This indicates that the analogies present in the test set are well characterized by embedding, and although the solution does not appear in the first position, it is still represented (in most cases) in the top ones. The embedding generated by this structure can therefore certainly be used as a better basis (since the analogies themselves characterize its goodness) for subsequent NLP problems, especially if the number and variety of analogies are increased.

In this case, however, the interest in using this network to generate better embedding is related solely to highlighting the limits of W2V.

One might actually expect that, given the relatively low number of modified vectors and the better position of the vectors obtained (relatively to the analogies), another embedding training (through the classic W2V scheme) will not excessively alter the advantages introduced by the second training. Instead, even if you set the learning rate to a very low value ($\eta = 10^{-4}$) and lock H by training Z alone for a certain number of epochs (in order to adapt the second part to the changes introduced in the first), further training gradually destroys all the advantages obtained (Fig. 7).

This attitude confirms that the W2V methodology always leads to a point of stability that depends on the dataset used, and that therefore the choice of a better starting point is not able to improve the final solution. On the other hand, the function that W2V tries to minimize is not very connected to the analogy test, which basically leads it not to recognize a better situation from that point of view.

5 Conclusions

In this work, we have analysed Word2Vec in the Skip-gram mode looking at different issues related to learning. Through a careful analysis, it has been noted that the model demonstrates better performance on the analogies mainly through the relationship it creates by contrasting the minimization of the loss function. The way in which the learning rate descends at each epoch, which goes in an opposite direction with respect to the classic objective of minimizing the loss, seems in fact to be fundamental in ensuring the creation of relationships between word vectors. This led to training the model by evaluating each epoch independently, in order to observe the results without being conditioned by the learning rate. The observation of learning as the number of epochs increases has also clearly shown that Word2Vec is unable to learn syntactic relationships, which instead seem to be mainly due to the link between the size of the training set and the available space. Furthermore, the quality of the embedding on the test set stabilizes on a maximum value, which therefore (regardless of computational and memory costs) should always be achieved if W2V is to be assessed against other methodologies.

We have also shown in our analysis how the various hyperparameters influence learning differently. The trend with varying negative sampling, for example, represents a further reason for the benefit of training over many epochs. Similarly, the analysis of the window size has shown that performance improves for higher values, and this happens even if the training is performed for a few epochs (compensating in some way the cost). On the contrary, the choice of the embedding size requires extreme care since a significant reduction in performance is due to both too small and too large embedding.

Finally, we have proposed to further train a given embedding directly on analogies. The use of an adequate structure, in fact, allows to obtain performances in the order of 97% by modifying only a few vectors. Changing only some vectors could result in better "semantic" embedding, which could be used as a basis for the resolution of further NLP problems. In any case, through this better solution, it is shown how W2V cannot maintain the advantage obtained. That is, the structure of W2V proves to be extremely dependent on the corpus, making semantic proximity only a side effect of its true objective function.

Funding Open access funding provided by Università degli Studi della Campania Luigi Vanvitelli within the CRUI-CARE Agreement.

References

1. Al-Matham RN, Al-Khalifa HS (2021) Synoextractor: a novel pipeline for Arabic synonym extraction using Word2Vec word embeddings. Complexity. https://doi.org/10.1155/2021/6627434
2. Almeida F, Xexéo G (2019) Word embeddings: a survey. arXiv:1901.09069
3. Altszyler E, Sigman M, Fernández Slezak D (2016) Comparative study of LSA versus Word2Vec embeddings in small corpora: a case study in dreams database. arXiv:1610.01520
4. Altszyler E, Ribeiro S, Sigman M, Fernández Slezak D (2017) The interpretation of dream meaning: resolving ambiguity using latent semantic analysis in a small corpus of text. Conscious Cognit. https://doi.org/10.1016/j.concog.2017.09.004
5. Balaneshin-Kordan S, Kotov A (2018) Deep neural architecture for multi-modal retrieval based on joint embedding space for text and images. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, Association for Computing Machinery, New York, NY, USA, WSDM'18, pp 28–36. https://doi.org/10.1145/3159652.3159735
6. Baroni M, Lenci A (2010) Distributional memory: a general framework for corpus-based semantics. Comput Ling 36:673–721. https://doi.org/10.1162/coli_a_00016
7. Baroni M, Dinu G, Kruszewski G (2014) Don't count, predict! a systematic comparison of context-counting versus context-predicting semantic vectors. In: 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014—Proceedings of the Conference, vol 1, pp 238–247. https://doi.org/10.3115/v1/P14-1023
8. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol 7700. Springer, Berlin. https://doi.org/10.1007/978-3-642-35289-8_26
9. Bengio Y, Ducharme R, Vincent P (2000) A neural probabilistic language model. J Mach Learn Res 3:932–938. https://doi.org/10.1162/153244303322533223
10. Devlin J, Chang MW, Lee K, Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
11. Di Gennaro G, Buonanno A, Di Girolamo A, Ospedale A, Palmieri FAN (2021a) Intent classification in question-answering using LSTM architectures. In: Progresses in Artificial Intelligence and Neural Systems, Smart Innovation, Systems and Technologies, vol 184. Springer, Singapore. https://doi.org/10.1007/978-981-15-5093-5_11
12. Di Gennaro G, Buonanno A, Di Girolamo A, Ospedale A, Palmieri FAN, Fedele G (2021b) An analysis of Word2Vec for the Italian language. In: Progresses in Artificial Intelligence and Neural Systems, Smart Innovation, Systems and Technologies, vol 184. Springer, Singapore. https://doi.org/10.1007/978-981-15-5093-5_13
13. Finley G, Farmer S, Pakhomov S (2017) What analogies reveal about word vectors and their compositionality. In: *SEM 2017—6th Joint Conference on Lexical and Computational Semantics, pp 1–11. https://doi.org/10.18653/v1/S17-1001
14. Firth JR (1957) A synopsis of linguistic theory 1930–55. Stud Ling Anal (Spec Vol Philol Soc) 1952–59:1–32
15. Gries S (2014) Frequency tables: tests, effect sizes, and explorations. In: Glynn D, Robinson JA (eds) Corpus Methods for Semantics Quantitative Studies in Polysemy and Synonymy, pp 365–389. https://doi.org/10.1075/hcp.43.14gri
16. Harris ZS (1954) Distributional structure. WORD 10(2–3):146–162. https://doi.org/10.1080/00437956.1954.11659520
17. Jang B, Kim I, Kim JW (2019) Word2vec convolutional neural networks for classification of news articles and tweets. PLoS ONE 14(8):1–20. https://doi.org/10.1371/journal.pone.0220976
18. Jianqiang Z, Xiaolin G, Xuejun Z (2018) Deep convolution neural networks for twitter sentiment analysis. IEEE Access 6:23253–23260. https://doi.org/10.1109/ACCESS.2017.2776930
19. Levy O, Goldberg Y (2014) Neural word embedding as implicit matrix factorization. Adv Neural Inf Process Syst 3:2177–2185
20. Levy O, Goldberg Y, Dagan I (2015) Improving distributional similarity with lessons learned from word embeddings. Trans Assoc Comput Ling 3:211–225. https://doi.org/10.1162/tacl_a_00134
21. Linzen T (2016) Issues in evaluating semantic spaces using word analogies. In: Proceedings of Evaluating Vector-Space Representations for NLP Workshop, pp 13–18. https://doi.org/10.18653/v1/W16-2503

22. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Proceedings of ICLR Workshop 2013
23. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Process Syst 26:1
24. Mikolov T, Yih WT, Zweig G (2013) Linguistic regularities in continuous space word representations. In: Proceedings of NAACL-HLT Workshop, pp 746–751
25. Miller GA, Charles WG (1991) Contextual correlates of semantic similarity. Lang Cogn Process 6(1):1–28. https://doi.org/10.1080/01690969108406936
26. Muhammad PF, Kusumaningrum R, Wibowo A (2021) Sentiment analysis using Word2Vec and long short-term memory (LSTM) for Indonesian hotel reviews. Proc Comput Sci 179:728–735, 5th International Conference on Computer Science and Computational Intelligence 2020. https://doi.org/10.1016/j.procs.2021.01.061
27. Naili M, Habacha A, Ben Ghezala H (2017) Comparative study of word embedding methods in topic segmentation. Proc Comput Sci 112:340–349. https://doi.org/10.1016/j.procs.2017.08.009
28. Nicolai G, Cherry C, Kondrak G (2015) Morpho-syntactic regularities in continuous word representations: a multilingual study. In: Proceedings of the NAACL-HLT Workshop, pp 129–134. https://doi.org/10.3115/v1/W15-1518
29. Pennington J, Socher R, Manning C (2014) Glove: global vectors for word representation, vol 14, pp 1532–1543. https://doi.org/10.3115/v1/D14-1162
30. Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language models are unsupervised multitask learners. In: OpenAi
31. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv:1609.04747
32. Schnabel T, Labutov I, Mimno D, Joachims T (2015) Evaluation methods for unsupervised word embeddings. In: Proceedings of Empirical Methods in Natural Language Conference, pp 298–307. https://doi.org/10.18653/v1/D15-1036
33. Turney P (2006) Similarity of semantic relations. Comput Ling 32:379–416. https://doi.org/10.1162/coli.2006.32.3.379
34. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. arXiv:1706.03762