Dauphine | PSL
UNIVERSITÉ PARIS

Large Language Models

Word2Vec for text classification

Lab session

Constant BOURDREZ constant.bourdrez@dauphine.eu

Submission Date : October 21, 2024

2024/2025 - Semester I

1 Introduction

Word2Vec is a contrastive representation learning model that learns dense vector representations, or embeddings, for words by capturing their contextual relationships in large corpora.

1.1 Preliminary Questions

Question 1: Similarity in Word Embeddings

Given a word w and a context c, we consider the similarity measure:

$$\sigma(c \cdot w) = \frac{1}{1 + e^{-(c \cdot w)}}$$

Since we want to minimize the loss function (1):

$$L(M(w, c)) = -\mathbb{1}_{c \in C^+} \log(\sigma(c \cdot w)) - \mathbb{1}_{c \in C^-} \log(1 - \sigma(c \cdot w)) \tag{1}$$

(a) For a positive context $c \in C^+$, we aim to maximize $\sigma(c \cdot w)$. This aligns with the Word2Vec objective, reducing the vector distance between the word and its context in the embedding space.

(b) For a negative context $c \in C^-$, we aim to minimize $\sigma(c \cdot w)$. This enhances the model's ability to distinguish a word from unrelated contexts, increasing the vector distance between the word and unrelated context in the embedding space.

Geometrically, this corresponds to adjusting the vector representations such that positive contexts are pulled closer to the word vector, while negative contexts are pushed away, in the high-dimensional embedding space.

Question 2: Chopra, Hadsell, and LeCun [1]

Contrastive Learning is, simply put:

(a) Contrastive Learning involves learning a similarity metric to differentiate between similar and dissimilar data pairs. It maps inputs into a space where the distance reflects semantic differences — the lower the distance, the more similar the data pair is, and vice versa.

On page 3 of [1], we find the expression (2):

$$L(W, (Y, X_1, X_2)_i) = (1 - Y) L_G(E_W(X_1, X_2)_i) + Y L_I(E_W(X_1, X_2)_i) \tag{2}$$

Here are the analogies that can be drawn with our setup:

(b) **Analog of Y**: A label indicating the nature of the context (positive or negative) for the word ($1 - Y$ being the opposite label).

(c) **Analog of E_W**: The neural network weights, or embeddings.

(d) **Analogs of L_G and L_I**: Loss functions for positive and negative pairs, influencing the similarity scores in the embedding learning process.

2 The Word2Vec model

One can find our implementation of Word2Vec in the `2_hw_word2vec_.ipynb` notebook, with all the parameters for training specified in the `config.yaml` file. We also used 0 to pad every borders in order to make the model robust when it comes to borders. We used different hyperparameters for training Word2Vec, which are listed below:

- `n_samples` = 1000 for faster training on the Word2Vec task; 5000 will be used for the classification task.
- `batch_size` = 500

- `n_epochs` = 20, as signs of overfitting started to appear.
- $R = [2, 5, 10]$ and $K = 4$. My intuition here was that if the context size is too large, the model may lose information, so the ratio $2RK$ must not be too high.
- `embedding_dim` = [128, 256] to utilize powers of two for computational efficiency.

2.1 Training the model

Below is an example of the trained model and its corresponding learning curves 1. We trained a total of six models to perform transfer learning for the classification task.

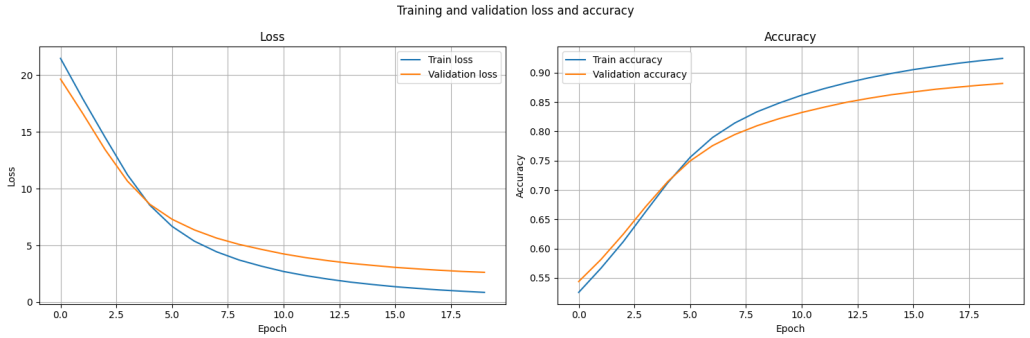

Figure 1: Learning curves for `embedding_dim` = 256 and $R = 5$. The plot illustrates how the model converges over time with these settings.

The performances of the six model on the word representation were pretty similar but led to the better performance. It also seems that bigger embedding dimension leads to poorer results when combined with the values of context size we used.

3 Transfer learning for the classification task

The implementation of the Classification Task can be found in the associated notebook 2_hw_classifier.ipynb. The training results of the Conv1D model initialized with the Word2Vec (w2v) embeddings, and with random ones, can be read on Figure 2.

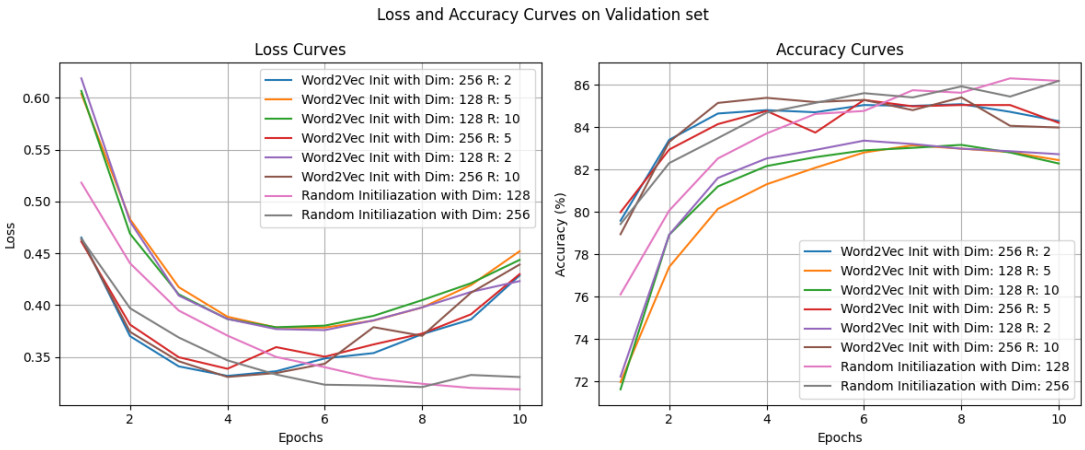

Figure 2: Difference of performance for all models on validation set

We used more data to train the classifier (*i.e* half of the dataset) in order to be sure to capture every phenomenon in this classification task. It can be seen that the pre-trained version does not improve accuracy

or loss, but allows the model to reach this type of performance much faster. It can be assumed that in more complex tasks, this type of initialisation can improve both the convergence time and the performance.

Then, if we compare only the training metrics one can notice this much faster convergence and performance on trainset on Figure 3.

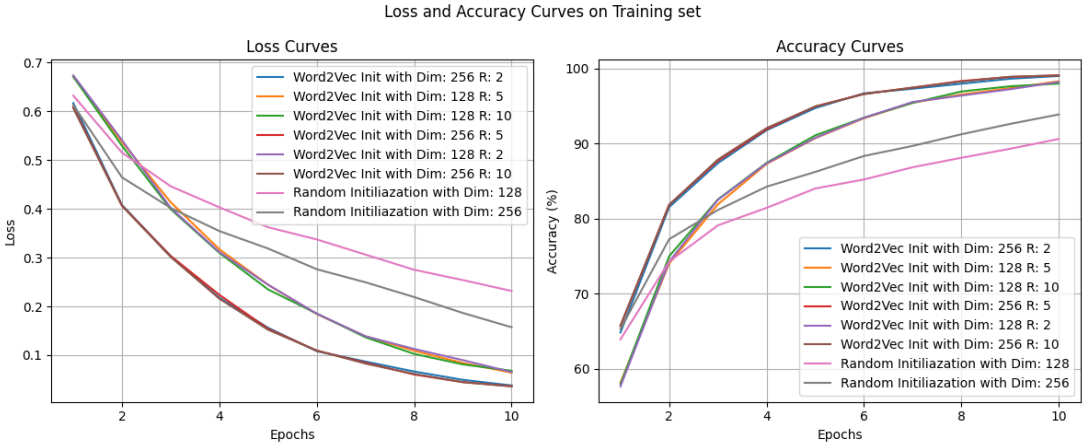

Figure 3: Metrics comparison for all models during training

3.1 Ablation study on the influence of R and the embedding dimension

We can also observe when fixing the embedding dimension, the effect of the context window size. It seems that for our case, the fewer the better. Nevertheless, the differences in performance is not that significant !

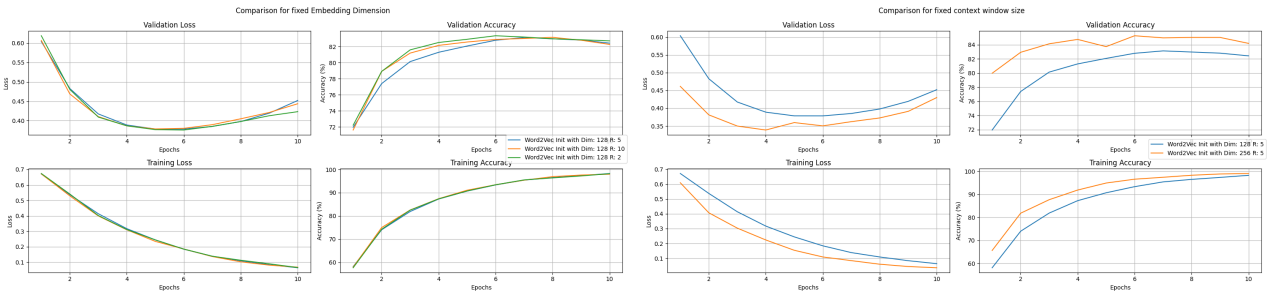

Figure 4: Comparison of the performances

On the other hand, it seems that increasing the embedding dimension leads to better results for all context window sizes tested.

4 Conclusion

From this project, I gained a deeper understanding of Word2Vec and its role in representation learning. I learned how to effectively train word embeddings by leveraging context and how contrastive learning techniques help separate meaningful relationships from random associations in a high-dimensional space. Additionally, the implementation and evaluation of Word2Vec enhanced my skills in handling large-scale language models and applying these embeddings to downstream tasks such as classification. Overall, this project provided insight into the practical applications of embedding models and the challenges involved in balancing model complexity with performance.

References

[1] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 539–546 vol. 1.