Homework 2 – Word2Vec

Course : Large Language Models

Université Paris Dauphine – PSL

Students : Louis Carron, Jules Roques

Program : Master 2 – Intelligence Artificielle, Systèmes et Données (IASD)

Supervisors :

M. Alexandre Allauzen

M. Florian Le Bronnec

Paris, le 10 octobre 2025

1 Introduction

2 Preliminary questions

1. (a) The context C^+ represents the word $\boldsymbol{w}$ surrounding in a sentence. To ensure that $\boldsymbol{w}$ carries coherent semantic meaning, we want to maximize its similarity $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$, with $\boldsymbol{c} \in C^+$.

   (b) On the contrary, C^- are random words with no particular semantic similarity with $\boldsymbol{w}$. Therefore we want to minimize the similarity $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ with $\boldsymbol{c} \in C^-$. Geometrically, the dot product measures how similar two vectors are by evaluating the cosine of the angle between them scaled by their norms. A high value will indicates that the two vectors are close neighbors in space, which in our application, signifies that their semantic meanings are close. And same inverse reasoning for a low similarity value.

2. (a) Contrastive learning is used in settings where the data only contains a small number of examples for a single category or a huge number of categories with some unknown during training phase. The principle is to learn a similarity metric which will compute a simple Euclidian distance on a mapping of the input data into a subspace encoding semantic meanings. It teaches a model to recognize similarity, as well as differences (hence the *contrastive* term), allowing it to compare new samples from categories it has never seen before.

   (b) In the article, Y is a label indicating whether a pair is *genuine* ($Y = 0$) or *impostor* ($Y = 1$) : Y indicates if a pair of images is not matching. In our setup, $\mathbf{1}_{c \in C^-}$ plays the same role : it acts as an indicator variable flagging pairs of words that are not in the same context.

   (c) In the article $E_W(X_1, X_2)$ is an energy function that computes the Euclidian distance between two embeddings produced by a convolutional neural network $G_W(X)$. In our setup, G_W corresponds to the embedding lookup table that maps a word index to its vector representation. Then, the similarity metric $\sigma(\boldsymbol{c} \cdot \boldsymbol{w})$ plays the role of the energy E_W between the inputs $\boldsymbol{w}$ and $\boldsymbol{c}$. The architectural difference arises from the distinct nature of the data : the original paper deals with images, whereas our application focuses on natural language. In Chopra et al. (2005), the model aims to reduce the Euclidean distance between embeddings of visually similar images, meaning closeness in position reflects visual similarity. In contrast, in NLP, the absolute position of a word vector has no semantic meaning, what matters is its direction. Therefore, we use cosine similarity to measure how aligned two embeddings are, capturing semantic relatedness through their orientation rather than spatial proximity.

   (d) Finally, the loss functions L_G, L_I are respectively analog to $-\log(\sigma(\boldsymbol{c} \cdot \boldsymbol{w}))$ and $-\log(1-\sigma(\boldsymbol{c} \cdot \boldsymbol{w}))$

3 Implementation

Summary

The first notebook, `hw2_word2vec.ipynb`, implements the full Word2Vec pipeline, structured into five modular parts : data preprocessing, context extraction, dataset creation, model definition, and training. During preprocessing, Wikipedia articles are tokenized using the `BertTokenizer` for consistent handling of subwords and punctuation. Zero-padding is applied at sentence borders to ensure constant context size, simplifying tensor batching in `PyTorch`. A fixed window of radius R is used to extract positive contexts. The PyTorch Dataset `WikiPediaWord2Vec` return words and their respective positive contexts. While negative context samples are uniformly drawn from the vocabulary with scaling factor K inside the `DataLoader`. The model is implemented as a `nn.Module` with two distinct embedding tables, one for target words and one for context words, trained by minimizing the binary cross-entropy loss using the Adam optimizer. In this first part hyperparameters (lr, B, R, K) were empirically chosen to balance computational efficiency and convergence stability.

3.1 Data preprocessing

1. An additional filtering step was applied. Empty text samples were removed, as well as Wikipedia balises, detected using a regular expression matching patterns such as `=== History ===`. It ensures that only meaningful content is kept for training, reducing noise. Before filtering, the dataset contained **36,718** entries ; after cleaning, it was reduced to **17,556**, a decrease of about **52%**.

2. In the context retrieval, a zero padding is used to deal with border issues. The zero id corresponds to the token [PAD] in `BertTokenizer`.
   Therefore, in the sentence *'The cat ate the mouse'*, for `R=2`, the context retreived for the first word '*The*' will be : $C^+ = [['PAD'], ['PAD'], 'cat', 'ate']$.
   This choice will ensure that the model learns the meaning of the token [PAD] as a transparent word, which does not carry any semantic information.

3.2 Model

10. We trained the model, using $B = 1024$, $lr = 1e^{-3}$ and 15000 samples, achieving smooth and stable convergence while avoiding overfitting within a few epochs.
11. Although a test split could have been added, we chose not to, since no explicit downstream evaluation is performed at this stage. Instead our model perform a validation step inside its training loop providing a sufficient estimate of the model's convergence and generalization behavior. The quality of the embeddings will instead be assessed later through the classification task in Section 4.
12. The function `save_model` was implemented inside the `Trainer` class.

Results

Figure 1 presents the training and validation loss curves. Note that, at this stage, R and K were chosen empirically to ensure stable convergence and satisfactory learning behavior. Their precise influence on the model's performance will be analyzed in detail later. The model converged to a final loss of approximately 0.1, indicating efficient learning and well-structured semantic representations, suitable for the downstream classification task described in Section 4.

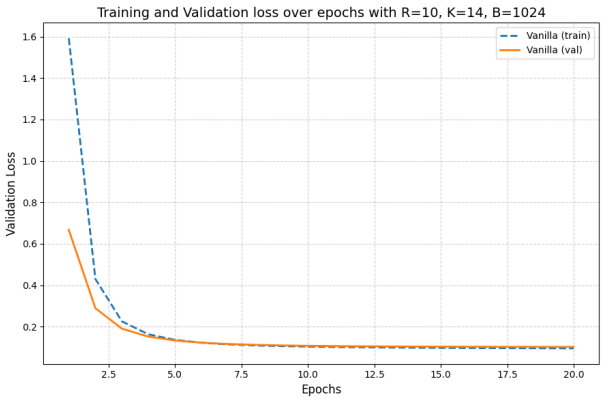


FIGURE 1 – Training and Validation loss over epochs for the Word2Vec models with $R = 10$, $K = 14$, $B = 1024$, $lr = 1e^{-3}$, 15000 samples on Google Collab (T4).

4 Classification task

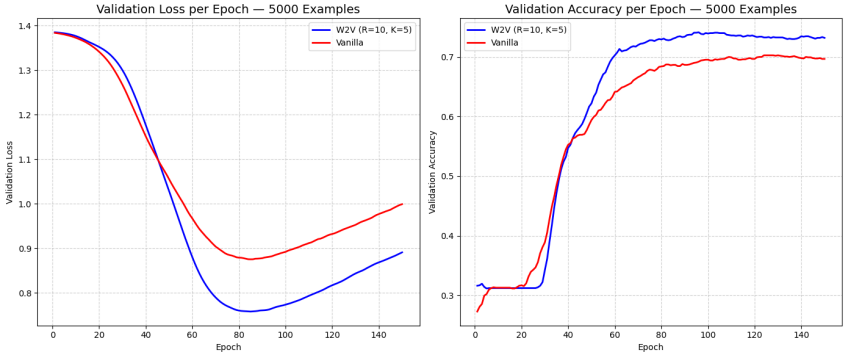
Summary

In this part, we evaluate the impact of using our pre-trained embedding table at the initialization of a classification task. For the classification task, we recover from the notebook of the last lab session `lab_2_attention_live.ipynb` classes and functions related to the model (`AttentionModel` and `ClassAttentionModel`), training, data loading and pre-processing. We used some native `torch` classes such as `nn.Linear` and `nn.Embedding` instead of the custom implementations provided in earlier courses. Then, we proceed step by step as suggested by the instructions. First, we pre-train our embedding weights with the *Word2Vec* model, for different configurations of hyper-parameters, and store them in *checkpoint* files. Then, we created a function `load_embedding_weights` (named `load_model` in the instructions) that retrieves weights given a file name, and use them to initialize the `ClassAttentionModel` embeddings. Note that our `load_embedding_weights` function allows whether to freeze the `Word2Vec` embedding for training or not. We chose not to freeze weights in our study. Finally, we trained the classifier with these different embedding initializations and compared the results with the Baseline model (Vanilla) with a random initialization.

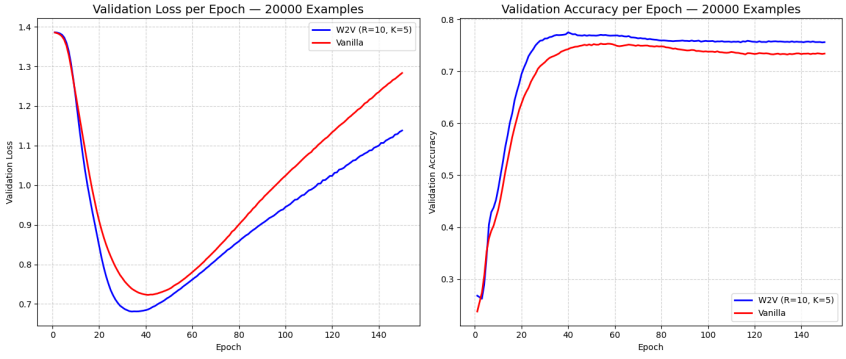
Results

For the first training of the Word2Vec classifier, we intuitively choose the value of the hyper parameters R and K. First, we assumed that performance would improve if the context window covered an entire sentence instead of a syntactic group. Since a typical sentence contains around 20 tokens, we set the radius to $R = 10$. The parameter K controls the number of negative samples and is essential for learning meaningful semantic contrasts. Therefore chose $K = 5$, which is equivalent to a negative context of almost 80% in proportion of the total context. The batch size used is $B = 256$, $lr = 1e^{-3}$ and the embedded dimension for Word2Vec and classification model is $d = 10$. The following Figure n°2, compares the validation loss and accuracy of the `ClassAttentionModel` with Word2Vec init and the Vanilla one (trained from scratch) for different number of examples in the dataset (Note that the dataset is splitted using 30% for validation).

Figure 2 illustrates the effectiveness of contrastive learning in both situations, with small and large amounts of labeled data. In both cases, models initialized with representations learned through the contrastive objective perform better than those trained from scratch. The advantage is particularly clear when only a small labeled dataset is available. Indeed, the pre-trained embeddings, having captured semantic information from large unlabeled data, transfer efficiently to the classification task. This initialization allows faster convergence, lower validation loss, and higher accuracy compared to a model relying solely on supervised learning. Contrastive pre-training therefore provides reusable and semantically structured representations that can be fine-tuned to surpass fully supervised approaches. In the next experiments, and because of the expensive computational time, the dataset will always contain 5000 samples.



(a) Loss and accuracy curves with 5000 examples.



(b) Loss and accuracy curves with 20000 examples.

FIGURE 2 – Validation loss and accuracy curves of the Vanilla (trained from scratch) and the Word2Vec classifier for respectivelly 5000 and 20000 samples in the Dataset.

Then, we conducted an ablation study on the parameters R and K of the Word2Vec model, and shown their impact on the classification task. An interesting behavior to evaluate regarding R is how the classifier performs when we try to acquire the semantic meaning of a syntactic group $R = 5$ ($|C^+| = 10$ tokens), a sentence $R = 10$ ($|C^+| = 20$ tokens) and a paragraph $R = 15$ ($|C^+| = 30$ tokens). The following Figure 3, shows that small value of R tends to produce the best validation loss and accuracy. In other

words, the most meaningful information to discriminate whether an article from WikiPedia belongs to the category World, Sports, Business or Sci/Tech is contained in only 10 tokens.

The observed result, where smaller context windows yield better validation loss and accuracy, is not surprising. Increasing R enlarges the semantic neighborhood and introduces less relevant co-occurrences, which tend to blur the discriminative information captured by the embeddings in Word2Vec. Since the classification task mainly depends on local lexical cues (e.g., topic-specific words), a limited context window is sufficient to encode the semantic distinctions required for accurate predictions. Larger contexts, while semantically richer, may tends to inject more noise than useful signal in this setting.

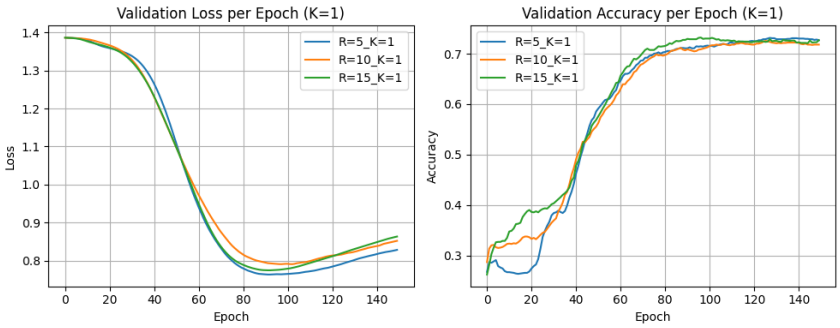


FIGURE 3 – Ablation study on values of R.

Reguarding the parameter K, we compared the classification performance for $K = 1$ (same proportion), $K = 2$ (∼60% of total context) and $K = 5$ (∼80%), shown in the above Figure 4. Increasing K, and therefore the proportion of negative samples relative to positive ones during Word2Vec training, consistently led to poorer classification results. In other words, when $K = 1$, the model maintains a good balance between learning from meaningful co-occurrences and contrasting with irrelevant ones. Once this ratio is exceeded, the optimization focuses too much on distinguishing random word pairs instead of reinforcing semantically relevant contexts, resulting in less informative latent representations for downstream tasks.

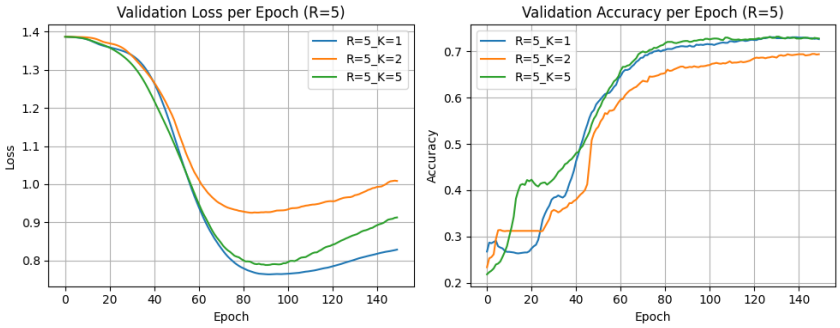


FIGURE 4 – Ablation study on values of K.

5 Conclusion

This project provided hands-on experience with contrastive learning through the full implementation of the Word2Vec model from scratch. It allowed us to understand how semantic relationships between words can emerge from simple co-occurrence statistics optimized through a contrastive objective.

By reusing the learned embeddings in a downstream classification task, we observed the benefit of pre-training over random initialization, especially when labeled data are scarce.

Finally, the ablation studies on the hyperparameters R and K highlighted how context size and negative sampling shape the semantic structure of the latent space, offering a concrete view of the trade-offs involved in representation learning.