Additional Experiments: Skip-Gram with Negative Sampling

Antoine Guiot, Arthur Claude, Armand Margerin

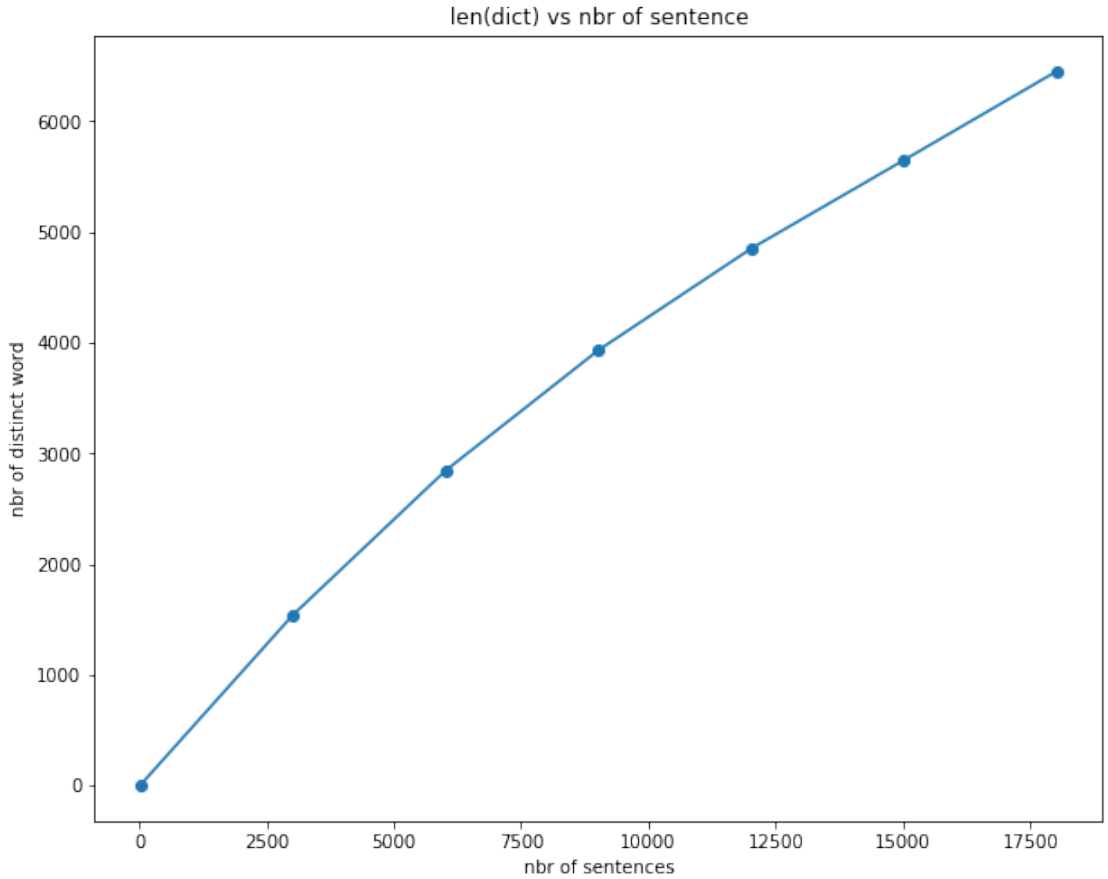
February 2020

1 Introduction

All the results presented here were obtained thanks to the notebook attached.
We will try to tune different hyperparameters of our model:
- try to run the model on different texts sizes
- try to run the model with different embedding sizes, window sizes ...

2 Data_set_size impact

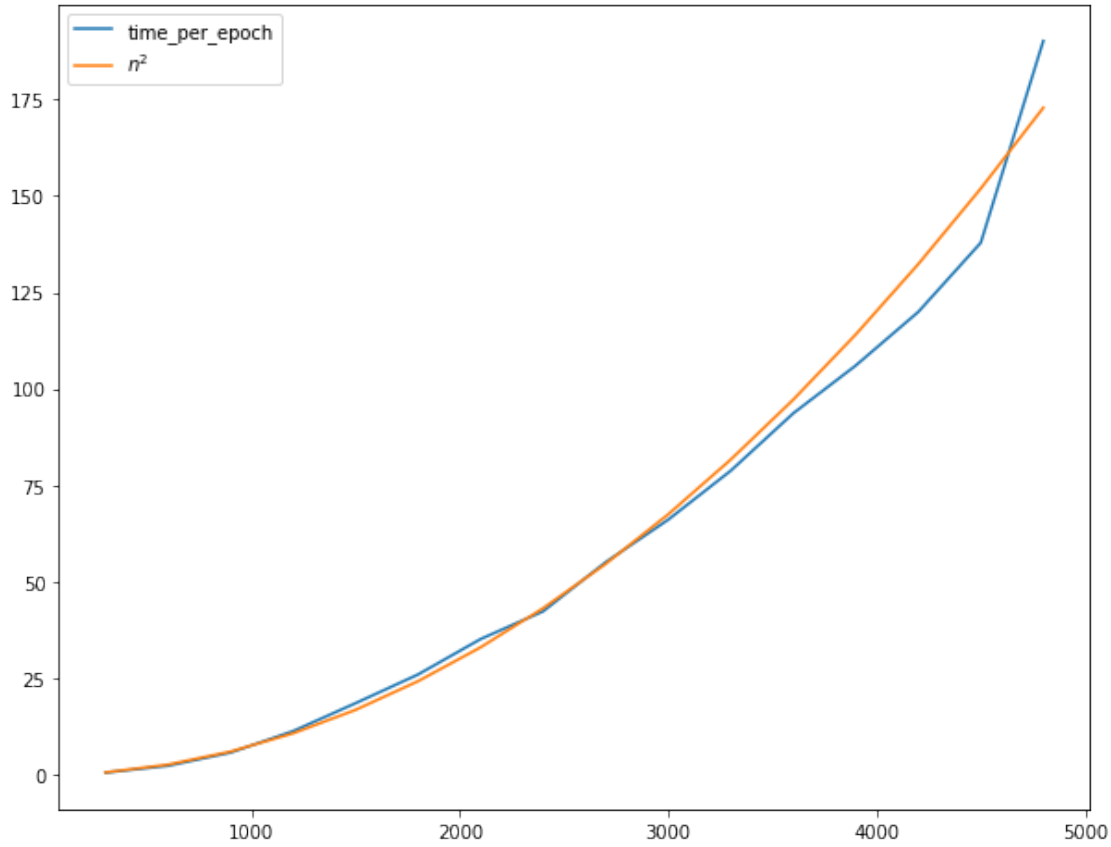
The first figure below shows the number of words in the dictionary as a function of the number of sentences.



As we can see above: if we want to have a good representation of the dictionary, we need to take a huge quantity of sentences.
But if we train our model on more than 10000 sentences, the training time becomes too long (see the next figure).

3 Time for training

We will train our model on different training_set size and see the impact on the training time:



As we can see the training time per epoch is not linear but might have a complexity n^2.

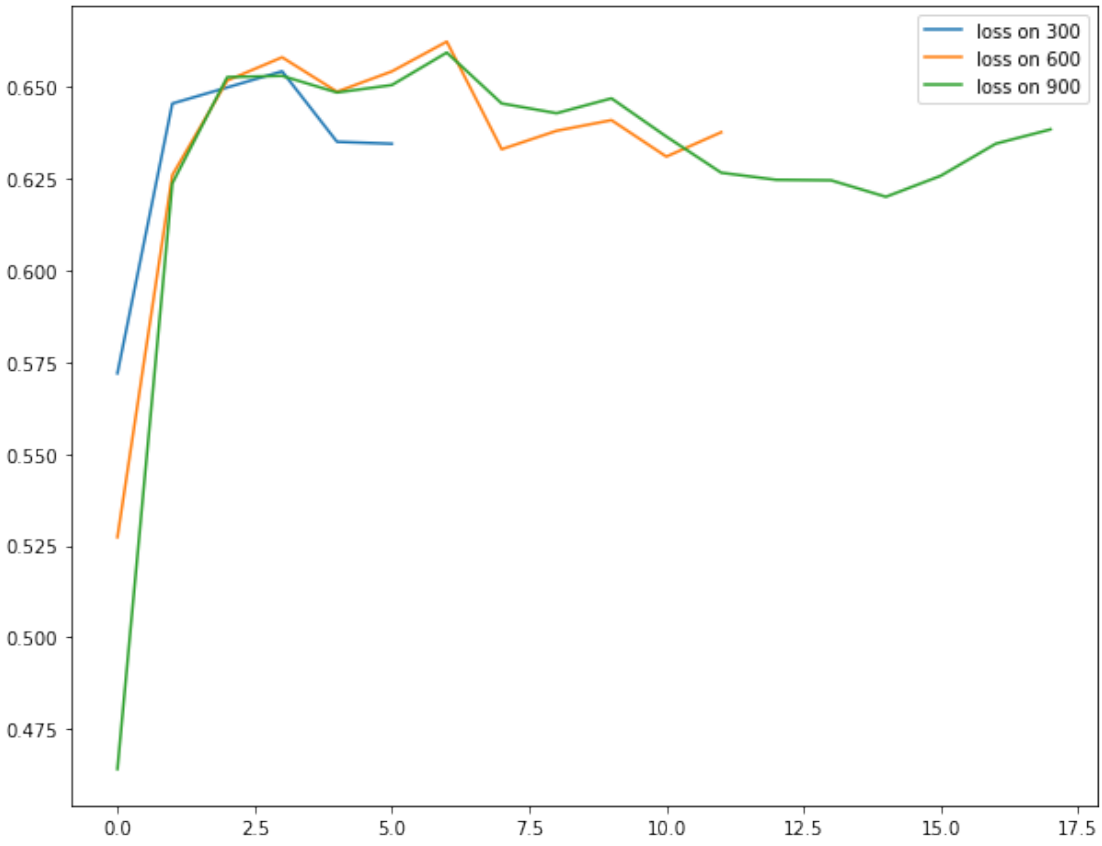
4 Loss

The loss is computed at each 100 words.
Example: for word_i and context word_j

$$loss(i,j) = \frac{1}{1+e^{-U_i V_j}}$$

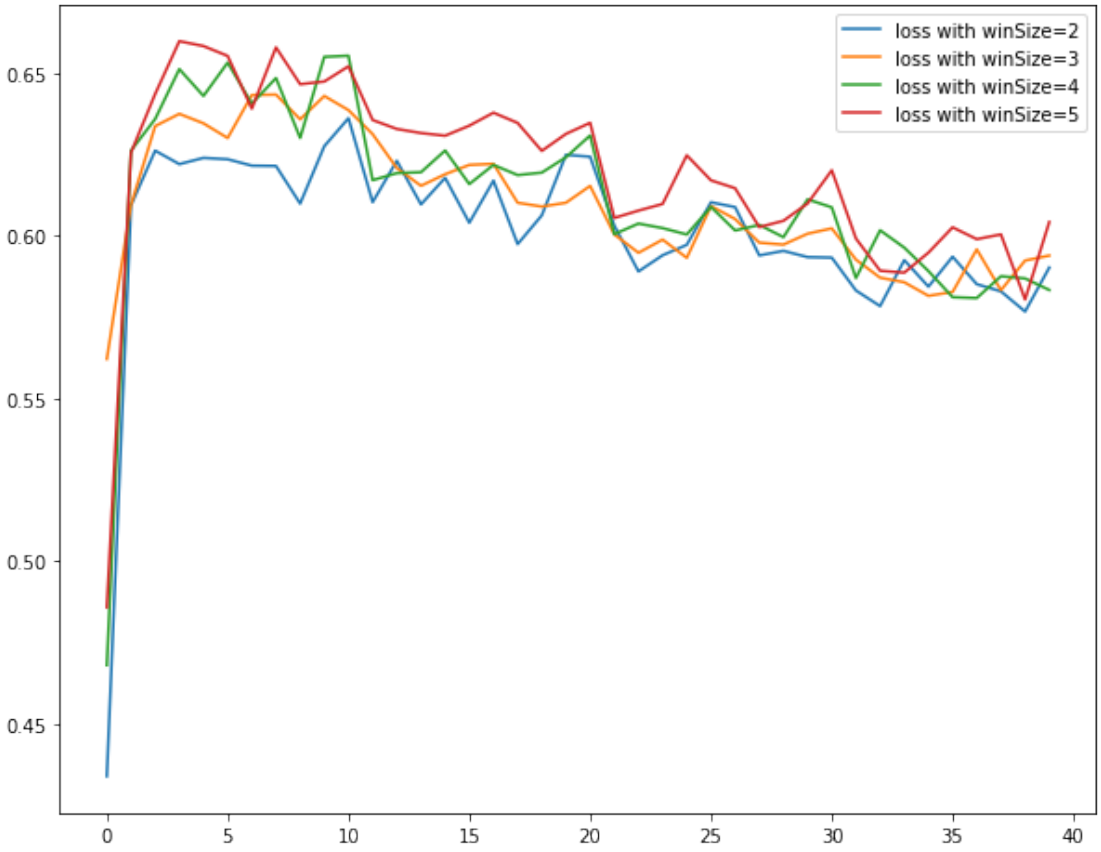
With V_j the context vector for the word j, and U_i the target vector for the word i.

During the training the loss should decrease: the next plot show us the loss function decreasing during the training for a model trained on 1000, 2000 and 3000 sentences.



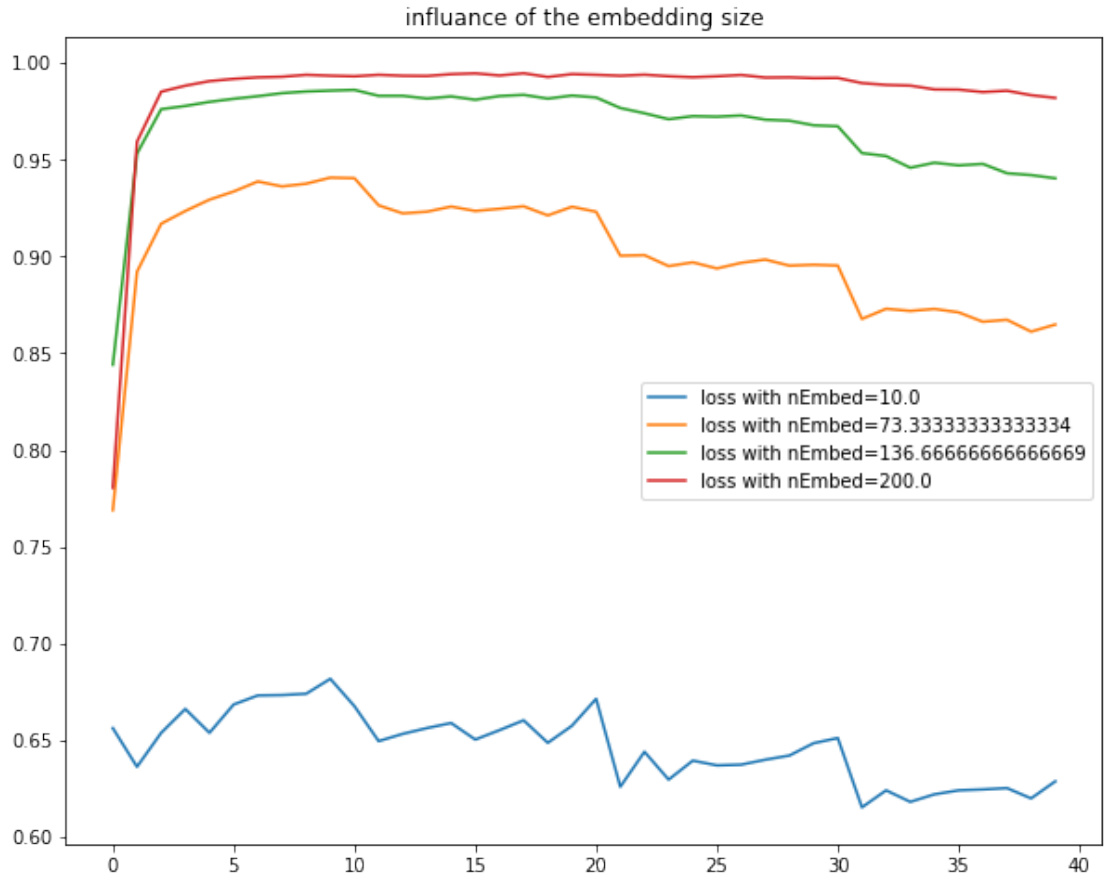
5 Hyperparameters influence

Our model contains some hyperparameters: winSize, negativerate, nEmbed. Here we will test the influence of these parameters on the loss function.



The winSize might have a little influence on the loss function. Indeed, the model seems to be better if the winSize is set as 2.

The negativeRate doesn't appear to have a big influence on the loss function.



On our example, the loss function seems to be better with a small embedding size (10). But here we trained our model on a very small data set containing 1000 sentences. In a real case, we would have a data set with more than 100 000 sentences and we would be able to train our model on many epochs.

6 Similarity

The similarity between two words is computed with the following equation:

$$s(word\_i, word\_j) = \frac{U_i U_j}{|U_i||U_j|}$$

At first, we will train a model on a 5000 sentences data set and then we will compute the similarity of two obviously similar words. For our example, we will compare the word “obama” and the word “president”.

[175]:
```
sg= SkipGram(sentences[0:5000])
sg.train(5)
```

[176]:
```
sg.similarity('obama','president')
```

[176]: 0.16917067050896503

This similarity should be close to 1 because these two words are often found together in a text. But here we train our model on 5000 sentences which is not enough to have a representative similarity.

7 Conclusion

In this notebook, we tried to tune some hyperparameters to improve the precision of our model (having the lowest loss function as possible). But because of the very long training time, we can not train our model on a big data set and on many epochs. So as a conclusion, it is difficult to know if our deductions on hyperparameters are really representative.