Lab exercise 1 - Deep Learning for NLP Simple bag-of-word classifier

[Sarah Aamiri][1] [Theo Deschamps-Berger][2] [Gérémy Hutin][3]

INTRODUCTION

The aim of this lab is to implement a bag-of-word classifier, i.e. a classifier that ignores the sequential structure of the sentence, with deep learning methods. The goal is to predict if a sentence is a positive or negative review of a movie.

I. METHOD

To train our model, we will use a dataset constructed from IMDB (positive and negative). We will use the first 5000 sentences in each review set.

We will use Pytorch tensors to represent the input data for our neural network. For that, we will create a dictionary that maps all the words of our training set into integers, in order to have tensors of integers representing input sentences indexing word embeddings.

After that, the NN retrieves the word embeddings from an embedding table and constructs the input of the MLP by summing over all embeddings (i.e. bag-of-word model).

Then, it goes through hidden layers (depending on our representation), and projects the hidden representation to the output space, which will be a scalar : (0) for a negative review and (+1) for a postive review.

II. RESULTS

We will train our model and observe the results with different hyper-parameters:
- Number of hidden layers
- Embedding dimension
- Pytorch optimizer

A. No hidden layer

We first experiment to train our Neural Network without any hidden layer, i.e. with applying on our input data only a linear function and a *sigmoid* activation function on the output, as we are doing a binary classification.

- With Embedding dimension = 4 and Optimizer = SGD:

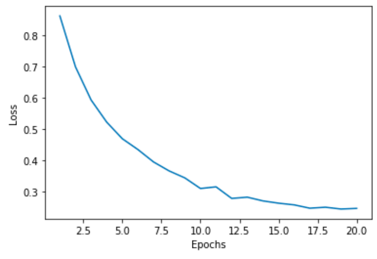

Fig. 1: The loss evolution on 20 epochs

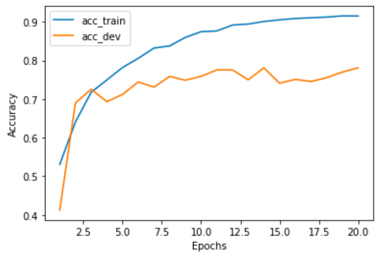

Fig. 2: The accuracy evolution on 20 epochs

We have an accuracy of 0.78 on the dev-set

- With Embedding dimension = 4 and Optimizer = Adam:

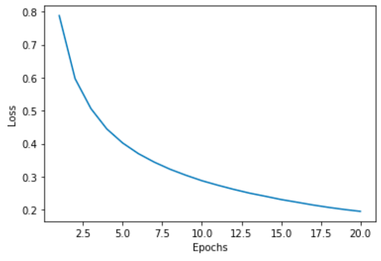

Fig. 3: The loss evolution on 20 epochs

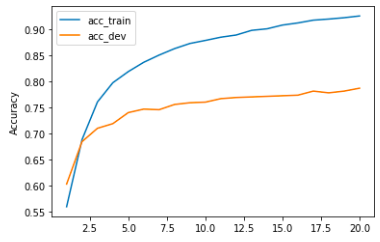

Fig. 4: The accuracy evolution on 20 epochs

We have an accuracy of 0.79 on the dev set

We chose to use the BCELoss function as a loss function, because the target is a scalar between 0 and 1, and we coded manually the output activation function *sigmoid*. (The BCEWithLogitsLoss includes the loss + the sigmoid function)

We used an adaptive learning rate method for our training loop, by reducing it while increasing the dev accuracy (approaching to the optimum).

Regarding the embedding dimension, it helps the Neural Network using more information about a word/sentence to get more accuracy for the prediction. By increasing the embedding

dimension, the accuracy should increase also. However, if we get too much information about a word/sentence (by using a big embedding dimension), this may lead us to over-fitting.
The Global Average pooling is the mean of all the vectors produced by the embedding step. The goal is to have a usable vector of fixed length for each sequence which will satisfy the input of our Neural Network.

Thus, the importance of a word in this vector depends on the sequence length it belongs.

After trying different hyper-parameters, we got the best results for *embedding dimension = 6* using the Adam optimizer: we obtain an **accuracy of 0.786 on the test set**, after 10 epochs.

B. One hidden layer

We add to our Neural Network 1 hidden layer. For that, we use the *tanh* activation function at the output of the hidden layer.

We observe for *embedding dimension = 6* and using the Adam optimizer:

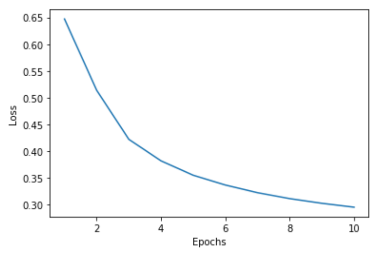

Fig. 5: The loss evolution on 10 epochs

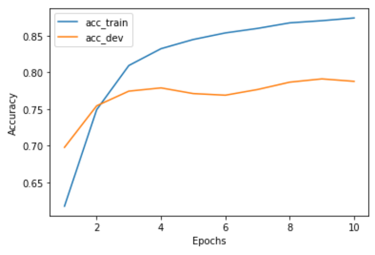

Fig. 6: The accuracy evolution on 10 epochs

We obtain an **accuracy of 0.786 on the test set**, after 10 epochs.

Conclusion

To conclude, we could say that whatever the model we used, our results were comparable and limited, which seems to indicate that the simple bag of word classifier is not enough to resolve this problematic.
The implementatiopn of a RNN instead of our model could lead to better results because we lose in our model, information about the order of our words in each sequence. The RNN type of model seems natural to treat NLP problems.