Deep Learning for Natural Language Processing

1. Monolingual embeddings

2. Multilingual word embeddings

Question:

We want to solve the following optimization problem:

$$W^* = argmin_{W \in O_d(\mathbb{R})} \|WX - Y\|_F$$

Which is the same as:

$$W^* = argmin_{W \in O_d(\mathbb{R})} \|WX - Y\|_F^{\,2}$$

Then:

$$W^* = argmin_{W \in O_d(\mathbb{R})} (WX - Y | WX - Y)$$
$$W^* = argmin_{W \in O_d(\mathbb{R})} \|X\|_F^{\,2} + \|Y\|_F^{\,2} - 2(WX|Y)$$

We can then transform the problem into:

$$W^* = argmax_{W \in O_d(\mathbb{R})} (WX|Y)$$
$$W^* = argmax_{W \in O_d(\mathbb{R})} (W|YX^T)$$

Using the singular value decomposition, we have:

$$YX^T = U\Sigma V^T, \textit{ with U and V two orthogonal matrixes}$$

Then, returning to our problem:

$$W^* = argmax_{W \in O_d(\mathbb{R})} (W|U\Sigma V^T)$$
$$W^* = argmax_{W \in O_d(\mathbb{R})} (U^T WV|\Sigma)$$

Using Cauchy-Schwarz inequality:

$$|(U^T WV|\Sigma)| \leq \|U^T WV\|_F \, \|\Sigma\|_F$$

The dot product is then maximized if $U^T WV = I_d$ (equality case in the Cauchy-Schwarz formula).

Finally, we have:

$$W^* = W = UI_d V^T$$
$$W^* = UV^T$$

3. Sentence classification with BoV

Question:

The two models, mean and IDF-weighted average, were tried. We got the following results, after training:

- Accuracy on the training set for the mean model: 0.4669
- Accuracy on the dev set for the mean model: 0.4024
- Accuracy on the training set for the IDF-weighted average model: 0.4434
- Accuracy on the dev set for the IDF-weighted average model: 0.3933

The parameters of the logistic regression were tuned using the dev set.

We can see that the best results we got in term of accuracy are when we use the mean model and not the IDF-weighted average one. We are then going to use the mean model to do the predictions and create the "logreg_bov_y_test_sst.txt" file.

4. Deep Learning models for classification

Question:

I used the categorical cross-entropy loss, which is the loss which provided me with the best results. I used this loss after a softmax activation.

The mathematical expression of this loss in our case is, for each example:

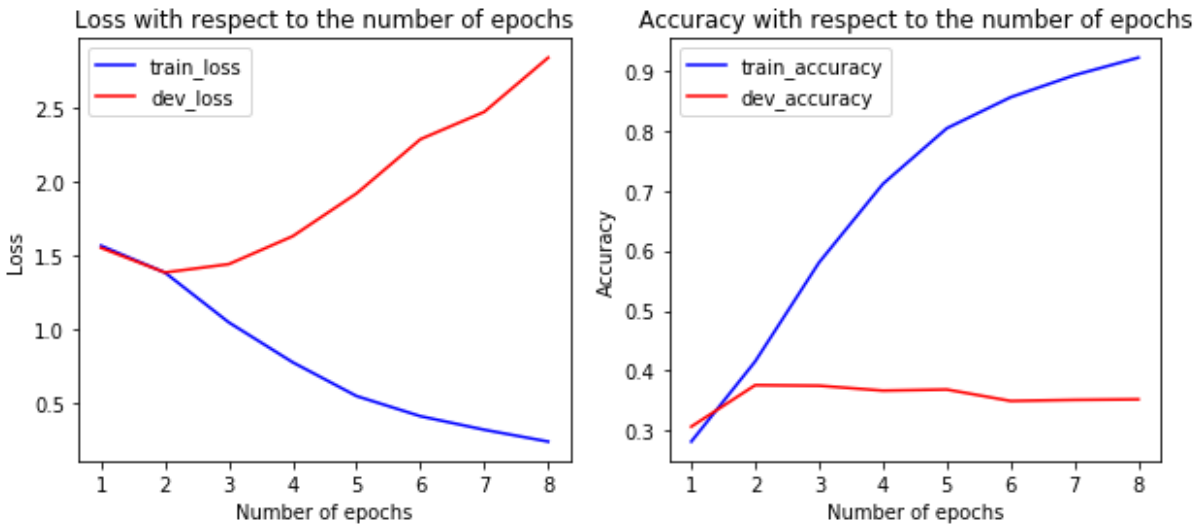
$$cce = -\sum_{i=1}^{5} t_i \, log(s_i)$$

In which:

- cce is the categorical cross-entropy.
- t_i is 1 only for one value of i. t_i is equal to 1 only for the i corresponding to the class the considered example belongs to. For the four other values of i, t_i is equal to 0.
- s_i is the output of the considered final neurons (the i^{th} neuron on the last layer), output just after the softmax activation.

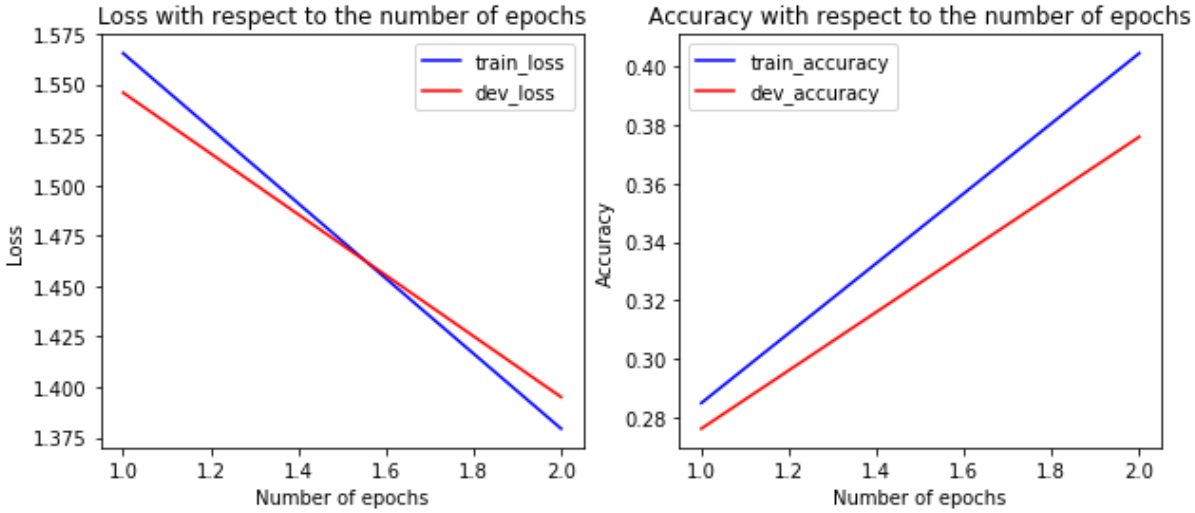
Question:

Here are the evolutions of train and dev results with respect to the number of epochs.



To produce this graph, we trained our network during 8 epochs. We can see that the network is quickly overfitting, with a increasing loss on the dev set whereas the loss on the train set is decreasing. After only two epochs, the accuracy on the dev set has reached its best and is quite stable: we are going to train our network for 2 epochs. We clearly have a network which quickly overfit.

The graphs for our training during two epochs are presented below.



Training the network for a third epoch would have decreased the quality of the results.

Question:

After different trials, the model that gave us the best results is the following.

The encoding of the sentences is not made with Keras but with our previous built classes Word2vec and BoV, which allows for a better representation of the sentences. These embeddings are determined thanks to much more examples (training is done on more examples). These embeddings also are more comprehensive as the embedding size is now 300 instead of 52 and takes into account the relative importance of a word thanks to the TF-IDF ponderation.

The network we are using is a 1D convolutional neural network composed of a convolutional layer and two fully connected layers. This network allows us to find patterns in the analyzed sentences which are relevant for classification: this network automatically build interesting features for our classification task. This is a different approach than LSTM and this is the main reason for using them here.

The classifications report on the dev set reports for our network an accuracy of 0.40, which is a bit better than what we had with the LSTM. The parameters of the network, like the number of filters, the kernel size, the number of neurons in the fully connected layer…, were tuned using the dev set.

Results on the test set will show if this CNN really is better than the LSTM previously built.