Deep Learning for Natural Language Processing

Arthur Claude

January 2020

1 Monolingual embeddings

See the notebook attached for code.

2 Multilingual embeddings

The goal is to find a mapping W that will map a source word space (e.g French) to a target word space (e.g English), such that the mapped source words are close to their translations in the target space.
We have :

$$
\begin{aligned}
W^* &= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmin}} \|WX - Y\|_F \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmin}} \|WX - Y\|_F^2 \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmin}} (\|WX\|_F^2 + \|Y\|_F^2 - 2\langle WX, Y\rangle_F) \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmin}} (\|X\|_F^2 + \|Y\|_F^2 - 2\langle WX, Y\rangle_F) \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \langle WX, Y\rangle_F \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \langle W, YX^T\rangle_F \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \langle W, SVD(YX^T)\rangle_F \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \langle W, U\Sigma V^T)\rangle_F \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \mathit{trace}((U\Sigma V^T)^T W) \\
&= \underset{W \in O_d(\mathbf{R})}{\operatorname{argmax}} \mathit{trace}(U^T W V \Sigma)
\end{aligned}
$$

We have, for $L \in O_d(\mathbf{R})$ and $\Sigma \in S_{n+}(\mathbf{R})$, $trace(L\Sigma) \leq trace(\Sigma)$. Thus, in our case, $trace(U^T W V \Sigma) \leq trace(\Sigma)$, with equality when $W = UV^T$. We have :

$$W^* = UV^T$$

See the notebook attached for code.

3 Sentence classification with BoW

See the notebook attached for code.

The results obtained with the logistic regression are :

```
Using the average of word vectors
Best C : 10
Dev score using average : 0.41689373297002724
Training score using average : 0.46594101123595505

Using the weighted-average of word vectors
Best C : 10
Dev score using weighted-average : 0.4141689373297003
Training score using weighted-average : 0.4708567415730337
```

As a bonus question, it were asked to try to improve the performance with another classifier. Unfortunately, I was not able to increase the score on the dev test. However, I managed to reach a very close score and finally propose this method: Support Vector Machine.

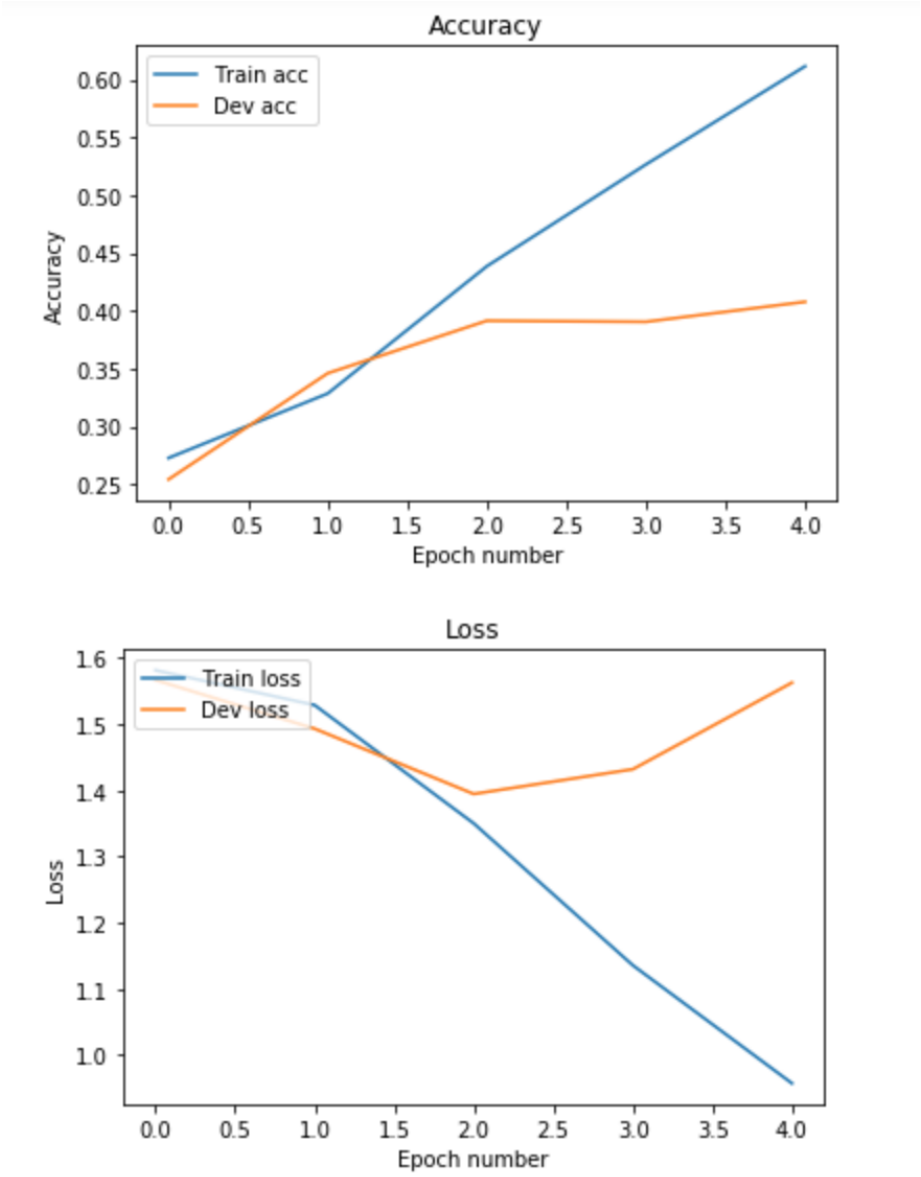
```
Using the average of word vectors
{'C': 1000, 'gamma': 0.01}
Dev score using average : 0.4150772025431426
Training score using average : 0.4785814606741573
```

4 Deep Learning models for classification

1) I used the Categorical Cross Entropy. For y_i the one hot encoded true label vector and p_i the predicted probability for each class, the loss function is :

$$L = -\sum_{i=0}^{4} y_i log(p_i)$$

2) We can plot the evolution of train / dev results :



We can see that from epoch 2, the classifier overfits the train set. I have tried some things to overcome this problem, unfortunately without success.

3) At this point, we were asked to try new models. I tried different models, but never got a better score. For exemple, I tried to use a Bidirectional LSTM. In fact, this kind of layers can extract more information from the sentences by going through these sentences in both ways. Even though I did not observe any improvement in the score, it is the result obtained on the test set with this model that I provide in my answers.

I also tried some convolutional neural networks, and combinations of recurrent and convolutional networks, without any significant results.

I wish I could have been able to improve my score and succeeded in eliminating the overfitting that systematically appears after a few epochs. I've encountered several problems during my research that I haven't been able to solve. For example, setting the value of "embed_dim" to 300 always causes an "AlreadyExistsError" error with my code. This stopped me from using the embeddings of the beginning of this project to set up a weight matrix for the (Bi-)LSTM layer. I think that this strategy could have helped me improve my model.