PAO Report for NLP : sentiment analysis

Robin Niel

January 14, 2019

Contents

Part I
SVM and Neural Network comparaison for sentiment analysis

1 Motivation

We want to compare the two main methods for sentiment analysis to make sure that the neural network I have developed is better at sentiment analysis than SVMs.

2 Experiences

To begin with, I started looking for specialised kernels for NLP. Sadly I couldn't find any ressources online about it, the people were recommanding using the linear kernel or the rbf one so that is the two I chose.

First we are gonna choose three models to test and choose from :

- SVM with linear kernel
- SVM with rbf kernel
- Neural network

The choosen neural network will be the best architecture I found during my speciality internship. It is summarised in the following array :

Layer name	Number of neurons	Activation function	Dropout	Filter size
Input layer	50	RELU	0.5	NA
Convolutional layer	64	RELU	0.5	10
LSTM layer	50		0.5	NA
Dense layer	50	RELU	0.5	NA
Dense layer	25	RELU	0.5	NA
Dense layer	2	Softmax	0	NA

Concerning the two SVMs, we need to find good hyperparameters in order to have a honest comparaison. To do so, I will under-sample the 2nd and the 3rd dataset in order to have shorter training time. So for the linear kernel, the only hyperparameter will be the C and for the rbf kernel, the C and the γ .

After finding the best hyperparameters, I will train the different models on the whole 2nd and 3rd dataset and then I will compare them on the same test set. After that I will test the three model on the first datasets in order to have a better idea about the generalization capacity of the different models.

3 Results

3.1 SVMs hyperparameters

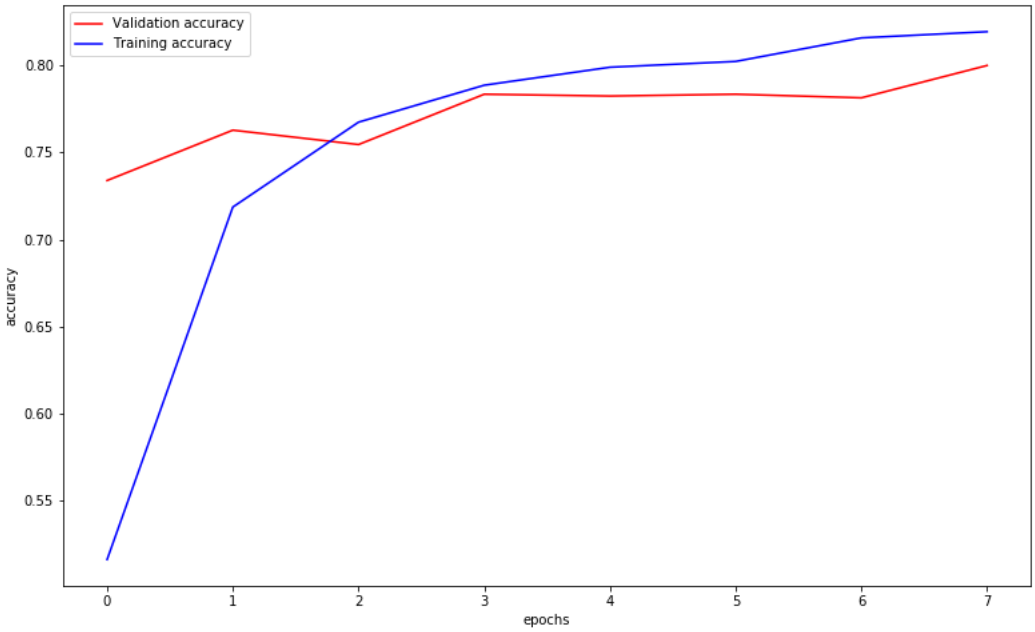
For the sub-sampling I took 2000 sentences. I found out that the best C for both SVMs was 1 and that the best γ was 0.1.

3.2 Second and third datasets comparaison

After training the three models, here are the results I obtained for the three models :

Model	Training time (mn)	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
Linear kernel	23,35	0,7017	0,734	0,681	0,699	0,7046
RBF kernel	85,30	0,5961	0,72	0,714	0,7	0,7113
Neural network	1,69	0,8044	0,841	0,816	0,786	0,8123

Following is the neural network train accuracy and loss.



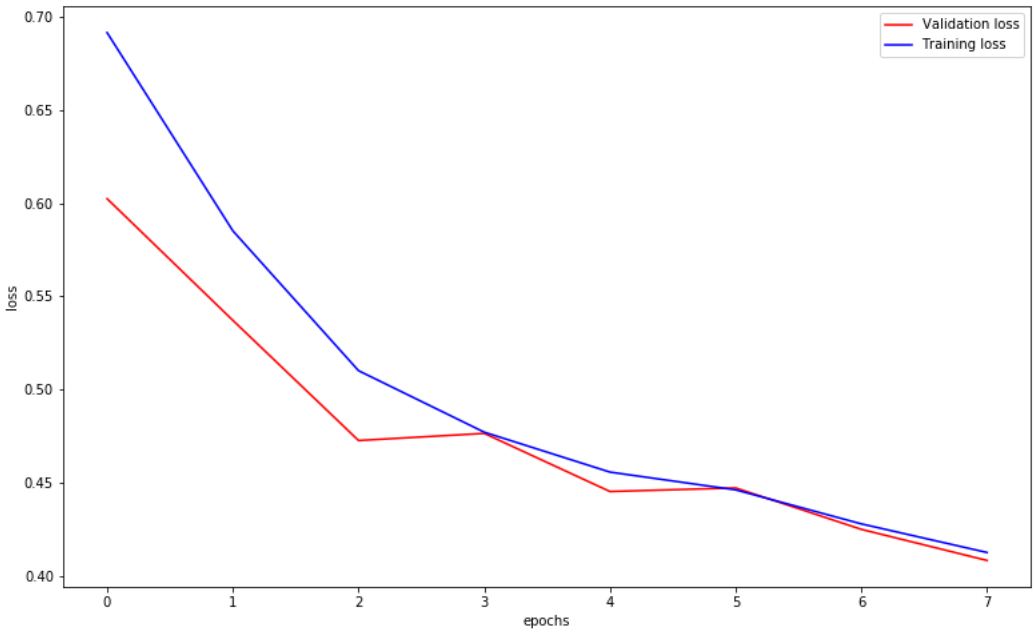


Figure 1: Training accuracy and loss

The first thing that we can notice is that the linear kernel performs better in validation but on a unknown dataset, the rbf kernel tends to perform a bit better. We can also notice that the neural network is performing way better than the two SVMs. We can therefore conclude that the neural network architecture is better than SVMs when it comes to sentiment analysis.

Part II
Experiments

4 Influence of dense layers and LSTM

In this par, we want to explore the influence of the different hyperparameters mainly the number of neurons in the dense layers or the number of LSTM. To do that we follow the same protocol as previously. Here are the results obtained :

Model	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
Dense {50,50}	0.7966	0.838	0.786	0.793	0.805
Dense {250,250}	0.7995	0.839	0.803	0.792	0.811
Dense {500,500}	0.8024	0.831	0.757	0.776	0.788
Dense {1024,1024}	0.8123	0.823	0.786	0.799	0.802
Dense {100,25}	0.8028	0.825	0.79	0.766	0.793
Dense {500,250}	0.8065	0.841	0.808	0.787	0.811
Dense {1000,500}	0.7986	0.828	0.773	0.767	0.789
Dense {250,250}, LSTM{150}	0.8053	0.808	0.765	0.775	0.783
Dense {250,250}, LSTM{100}	0.8115	0.815	0.757	0.781	0.784
Dense{250,250}, LSTM{50}	0.8094	0.819	0.785	0.79	0.798

For the number of neurons in the dense layers, we can say that the best configuration is the 500,250 or the 250 250.

For the LSTM part, we can observe a decrease in accuracy as the number of LSTM goes higher than 50. The optimal number of LSTM seems to be 50.

5 Negation propagation

5.1 Experiment

One of the current problem of the netork is the handling of negation. Indeed, the network doesn't pick that a "not" is inverting the meaning of a word. For example, "not good" is supposed to mean ~"good" but currently, the network has trouble predicting sentences with not. For example :

```
It was not good at all
 Negative        Positive
[0.14867947 0.85132056]
```

```
It was not bad at all. I'd go back there again
 Negative        Positive
[1.0000000e+00 6.1480035e-12]
```

Figure 2: False negative and positive

In order to do that, when a "not" is encountered in a sentence, we delete it and then propagate it to every words after until we find a punctuation, that is to say {",",".","!","?",":"}.

Here is a quick example of the negation propagation :

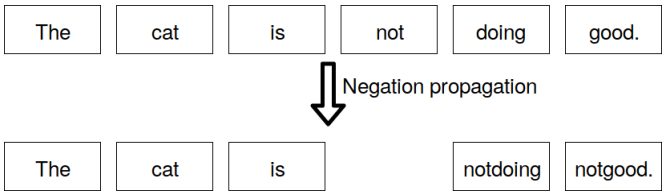


Figure 3: Negation propagation

This will form new words such as "notbad". The problem with that is that the word2vec model I was using, the GoogleNews one, canno't be trained again since we only have the vectors. So I need to train new word2Vec model on the amazon dataset at the following adress : `http://jmcauley.ucsd.edu/data/amazon/`. I will first train it on the Movies and Electronics dataset. I will also need to train a second one where the negation propagation is not used in order to compare the performance of both.

If the negation propagation if usefull, we should observe an increase in the accuracy following the protocol. To test this hypothesis, we will take the Dense{250,250}, LSTM{50} model and compare the result with and without the negation propagation.

5.2 Results

After training the new Word2Vec model and apply it on the data with the following neural architecture of neural network : Dense{250,250}, LSTM{50}, here is what is obtained after the experiment :

Word2Vec	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
"not" sentences percentage	0.6 %	8,3 %	10,7 %	12 %	10,3 %
GoogleNews	0.7928	0.832	0.825	0.825	0.827
Custom without not propagation	0.7804	0.837	0.775	0.795	0.802
Custom, not propagation	0.7837	0.809	0.834	0.773	0.805
Custom + embedding, not propagation	0.7771	0.816	0.852	0.78	0.816

We can observe a notable increase in accuracy for the unknown datasets. It's also better than the vanilla GoogleNews word2vec model.

6 The batch size issue

I did not focused on the batch size before because it is said to be an hyperparameter that is not that important. However, different colleagues told me that in the case of NLP, that was really important. So for the google news word2Vec and the "not propagation" model, I decided to plot the test accuracy and loss in function of the batch size.

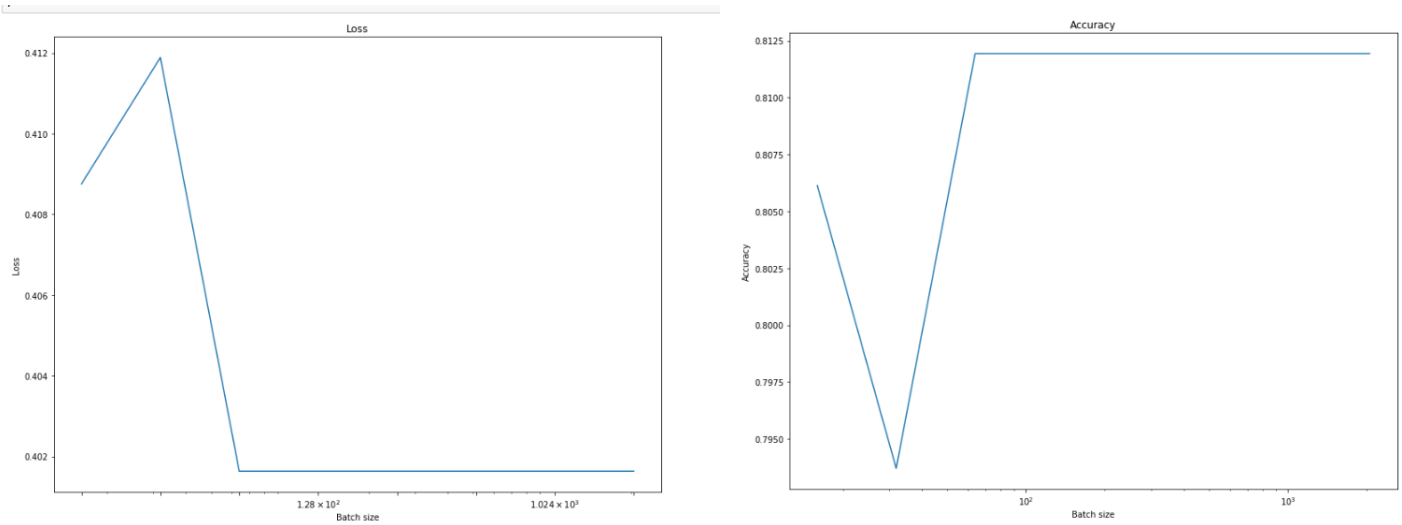


Figure 4: Accuracy and loss for the GoogleNews Word2vec model

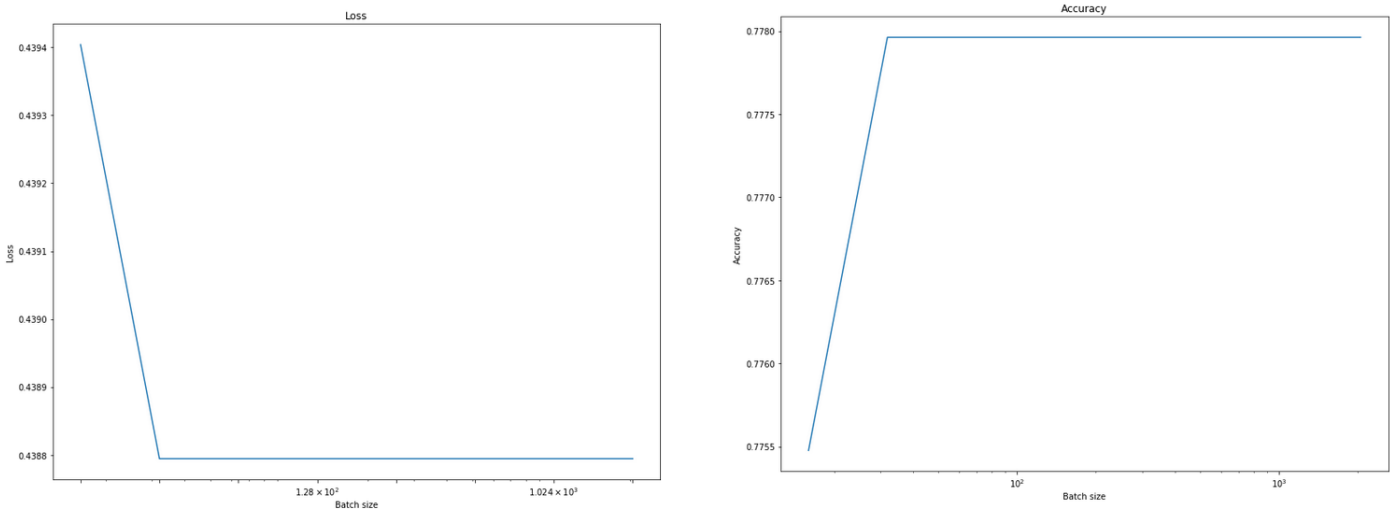


Figure 5: Accuracy and loss for the custom Word2vec model

We can observe a really big difference for the Google news model. The difference is quite small for the custom Word2Vec model. We can observe that the accuracy and loss is converging from a given point. We can conclude that the batch size must be big enough when comparing models. So for the next experiments, we will choose a batch size of 1028.

7 Training on amazon dataset

For this work, we were training only on 10000~ number of examples. In order to allow the network to modify its embedding layer, I choose to train it over two datasets from amazon, one with 1 689 188 data and the other with 1 697 533 for a total of 3.3~ M data. This data is composed of reviews rated from 1 to 5 stars. In order to apply the same test protocol we need to "binarize" the data, that is to say transform the y. To do that, I choose to classify the rating of 4 and 5 stars as positive and the 1 and 2 stars as negative. The 3 stars reviews can be keeped for later as a neutral class.

The expected result is the embedding model getting better than the one with the external Google News Word2vec model. Here are the obtained results. I also put the previous embedding line in order to compare them.

Word2Vec	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
GoogleNews	NA	0.5	0.5	0.5	0.5
Custom + embedding, not propagation	*0.7771*	*0.816*	*0.852*	*0.78*	*0.816*
Custom + embedding, not propagation	NA	0.84	0.866	0.753	0.822

As expected we can see an improvement in the total accuracy (0.816 -> 0.822). The accuracy has gotten better for both the IMDB and Amazon dataset although a small decrease in accuracy can be seen for the Yelp dataset.