Project Title : Sentiment Analysis

(Option 1 of NLP-based Projects)

Sentiment Analysis - Find the sentiment of a Tweet, whether it is positive or negative (And here, because of the database, neutral)

Link:

Project link (Github/ Gitlab): https://github.com/Lenarrateur/AiTweetSentiment

Team Members:

Student Name	Student ID	Contribution in the project
Nicolas MERLIN	73204	Model score, hyperparameter optimization and model save
Teanie Kim MIOSOTIS	73174	Data analysis, cleaning the dataset and split into train/split
Gaétan PETER	73327	Conversion to handle 3 types (neutral positive negative) and training

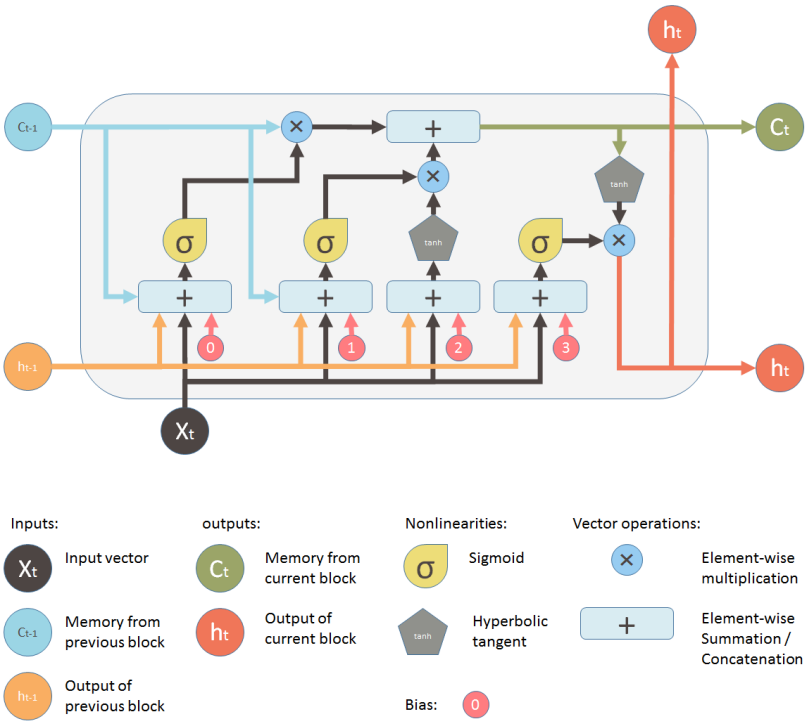
Model Architecture:

Explain the following in this page:

1. What model architecture have you selected

   The chosen model architecture is a Long Short-Term Memory (LSTM) neural network for sentiment analysis.

2. Put a diagram of the architecture (Schematic)



3. How many layers does the model have?

The model has four layers: The Embedding Layer, the LSTM Layer, the Dropout Layer and the Fully-connected (Linear) Output Layer

4. How many hidden layers and neurons do you have?

The embedding layer has 64 neurons, which correspond to the size of the output vectors, the LSTM layer is composed of two layers of 256 neurons (so 512 neurons total) and the Output Layer has three neurons (for the three classes). So a total of 579 neurons.

5. Have you changed the model architecture to improve performance?

Yes, we've tried modifying hyperparameters such as "embedding_dim" or "n_layers" to improve model performance. We'll come back to this step and the various results we obtained later in this report.

Dataset Description:

Explain the following in this page:

1. Details of the dataset (number of classes/ instances/ Images etc)

```
tweet.head(10)
```

	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative
5	28b57f3990	http://www.dotheboucy.com/smf - some shameles...	http://www.dotheboucy.com/smf - some shameles...	neutral
6	6e0c6d75b1	2am feedings for the baby are fun when he is a...	fun	positive
7	50e14c0bb8	Soooo high	Soooo high	neutral
8	e050245fbd	Both of you	Both of you	neutral
9	fc2cbefa9d	Journey!? Wow... u just became cooler. hehe....	Wow... u just became cooler.	positive

The dataset has 27.5k rows and 4 columns.

The columns are :

-textID (a unique ID for each piece of text)

-text (the text of the tweet)

-selected_text (a piece of text from the column text)

-sentiment (the general sentiment of the tweet, can be positive, negative or neutral)

Each row contains the text of a tweet and a sentiment label. In the training set you are provided with a word or phrase drawn from the tweet (selected_text) that encapsulates the provided sentiment.

2. What kind train / test split has been used

The type of train/test split used is the Holdout method. The dataset is divided into two subsets: training set (text) and testing set (sentiment)

3. What kind of data augmentation have been used

The type of data augmentation used is Text Data Augmentation by removing all non-word characters except numbers and letters, replacing all runs of whitespaces with no space and replacing digits with no space.

Methodology:

Explain the following in this page:

1. Training parameters, number of epochs, learning rate etc.

The model has been trained on 2 epochs over 1 parameter, but each epoch does around 450 steps with a batch size of 50, which explains why we decided to train our model “only” over two epochs. The learning rate is pretty slow, but it managed to increase up to 62% (we were at 54% after the first epoch, so it seems we’ve reached the limit of our model (maybe it could go up to 65% with 8 more epochs).

2. Loss function used for training

Here, we are using the `nn.CrossEntropyLoss()`. It basically compares the probability he gave for each output and compares it to the true answer, trying to minimize the loss (and so increase his accuracy).

3. Changes is parameter affecting the model results

To test the effectiveness of a parameter, we can't afford to run several versions of the model to completion (with different parameters) and then compare their scores: it's not realistic given the time available.

That's why we're only going to run the model for 50 steps (about 30 minutes) before comparing it with the other versions. This is long enough for us to notice any effect on the model, and to keep us on schedule.

Impact of the `embedding_dim`

The `embedding_dim` is …

The default value is 64. We also tested 128 and 32. Here are the different results we obtained:

```
Epoch: 1/2... Step: 1... Loss: 1.103383... Acc: 0.333333
Epoch: 1/2... Step: 2... Loss: 1.097692... Acc: 0.395833
Epoch: 1/2... Step: 3... Loss: 1.090037... Acc: 0.375000
Epoch: 1/2... Step: 4... Loss: 1.105841... Acc: 0.229167
Epoch: 1/2... Step: 5... Loss: 1.066458... Acc: 0.437500
Epoch: 1/2... Step: 6... Loss: 1.073151... Acc: 0.458333
Epoch: 1/2... Step: 7... Loss: 1.041626... Acc: 0.500000
Epoch: 1/2... Step: 8... Loss: 1.084316... Acc: 0.437500
Epoch: 1/2... Step: 9... Loss: 1.069708... Acc: 0.437500
Epoch: 1/2... Step: 10... Loss: 1.091720... Acc: 0.395833
Epoch: 1/2... Step: 11... Loss: 1.049131... Acc: 0.500000
Epoch: 1/2... Step: 12... Loss: 1.153289... Acc: 0.312500
Epoch: 1/2... Step: 13... Loss: 1.154297... Acc: 0.312500
Epoch: 1/2... Step: 14... Loss: 1.049983... Acc: 0.500000
Epoch: 1/2... Step: 15... Loss: 1.081127... Acc: 0.437500
Epoch: 1/2... Step: 16... Loss: 1.100783... Acc: 0.333333
Epoch: 1/2... Step: 17... Loss: 1.100928... Acc: 0.312500
Epoch: 1/2... Step: 18... Loss: 1.078407... Acc: 0.416667
Epoch: 1/2... Step: 19... Loss: 1.069827... Acc: 0.416667
Epoch: 1/2... Step: 20... Loss: 1.074242... Acc: 0.520833
Epoch: 1/2... Step: 21... Loss: 1.103835... Acc: 0.375000
Epoch: 1/2... Step: 22... Loss: 1.103709... Acc: 0.312500
Epoch: 1/2... Step: 23... Loss: 1.112499... Acc: 0.354167
Epoch: 1/2... Step: 24... Loss: 1.090071... Acc: 0.333333
Epoch: 1/2... Step: 25... Loss: 1.100960... Acc: 0.270833
Epoch: 1/2... Step: 26... Loss: 1.096678... Acc: 0.354167
Epoch: 1/2... Step: 27... Loss: 1.091601... Acc: 0.375000
Epoch: 1/2... Step: 28... Loss: 1.088313... Acc: 0.375000
Epoch: 1/2... Step: 29... Loss: 1.098487... Acc: 0.312500
Epoch: 1/2... Step: 30... Loss: 1.068975... Acc: 0.500000
Epoch: 1/2... Step: 31... Loss: 1.075597... Acc: 0.437500
Epoch: 1/2... Step: 32... Loss: 1.063829... Acc: 0.541667
Epoch: 1/2... Step: 33... Loss: 1.082570... Acc: 0.437500
Epoch: 1/2... Step: 34... Loss: 1.070205... Acc: 0.437500
Epoch: 1/2... Step: 35... Loss: 1.078566... Acc: 0.375000
Epoch: 1/2... Step: 36... Loss: 1.105114... Acc: 0.333333
Epoch: 1/2... Step: 37... Loss: 1.075297... Acc: 0.416667
Epoch: 1/2... Step: 38... Loss: 1.084491... Acc: 0.375000
Epoch: 1/2... Step: 39... Loss: 1.083589... Acc: 0.354167
Epoch: 1/2... Step: 40... Loss: 1.083729... Acc: 0.416667
Epoch: 1/2... Step: 41... Loss: 1.081327... Acc: 0.395833
Epoch: 1/2... Step: 42... Loss: 1.068139... Acc: 0.437500
Epoch: 1/2... Step: 43... Loss: 1.066267... Acc: 0.520833
Epoch: 1/2... Step: 44... Loss: 1.048187... Acc: 0.500000
Epoch: 1/2... Step: 45... Loss: 1.075000... Acc: 0.500000
Epoch: 1/2... Step: 46... Loss: 1.075988... Acc: 0.416667
Epoch: 1/2... Step: 47... Loss: 1.097630... Acc: 0.354167
Epoch: 1/2... Step: 48... Loss: 1.060880... Acc: 0.500000
Epoch: 1/2... Step: 49... Loss: 1.029341... Acc: 0.520833
Epoch: 1/2... Step: 50... Loss: 1.057473... Acc: 0.458333
Epoch: 1/2... Step: 51... Loss: 1.054676... Acc: 0.437500
```

```
Test Loss: 0.604... Test Acc: 46.534%
```

For 128 and for 32 :

```
Epoch: 1/2... Step: 1... Loss: 1.098233... Acc: 0.229167
Epoch: 1/2... Step: 2... Loss: 1.100479... Acc: 0.229167
Epoch: 1/2... Step: 3... Loss: 1.098214... Acc: 0.291667
Epoch: 1/2... Step: 4... Loss: 1.092488... Acc: 0.416667
Epoch: 1/2... Step: 5... Loss: 1.092403... Acc: 0.333333
Epoch: 1/2... Step: 6... Loss: 1.062313... Acc: 0.562500
Epoch: 1/2... Step: 7... Loss: 1.066584... Acc: 0.520833
Epoch: 1/2... Step: 8... Loss: 1.074522... Acc: 0.416667
Epoch: 1/2... Step: 9... Loss: 1.074466... Acc: 0.458333
Epoch: 1/2... Step: 10... Loss: 1.137569... Acc: 0.354167
Epoch: 1/2... Step: 11... Loss: 1.112069... Acc: 0.395833
Epoch: 1/2... Step: 12... Loss: 1.084533... Acc: 0.416667
Epoch: 1/2... Step: 13... Loss: 1.145055... Acc: 0.250000
Epoch: 1/2... Step: 14... Loss: 1.075398... Acc: 0.354167
Epoch: 1/2... Step: 15... Loss: 1.077097... Acc: 0.437500
Epoch: 1/2... Step: 16... Loss: 1.072913... Acc: 0.479167
Epoch: 1/2... Step: 17... Loss: 1.086637... Acc: 0.395833
Epoch: 1/2... Step: 18... Loss: 1.070043... Acc: 0.416667
Epoch: 1/2... Step: 19... Loss: 1.095558... Acc: 0.354167
Epoch: 1/2... Step: 20... Loss: 1.098894... Acc: 0.395833
Epoch: 1/2... Step: 21... Loss: 1.087203... Acc: 0.416667
Epoch: 1/2... Step: 22... Loss: 1.089260... Acc: 0.354167
Epoch: 1/2... Step: 23... Loss: 1.088167... Acc: 0.395833
Epoch: 1/2... Step: 24... Loss: 1.070739... Acc: 0.416667
Epoch: 1/2... Step: 25... Loss: 1.084578... Acc: 0.395833
Epoch: 1/2... Step: 26... Loss: 1.079987... Acc: 0.395833
Epoch: 1/2... Step: 27... Loss: 1.065108... Acc: 0.500000
Epoch: 1/2... Step: 28... Loss: 1.124330... Acc: 0.270833
Epoch: 1/2... Step: 29... Loss: 1.073479... Acc: 0.458333
Epoch: 1/2... Step: 30... Loss: 1.101217... Acc: 0.375000
Epoch: 1/2... Step: 31... Loss: 1.070801... Acc: 0.354167
Epoch: 1/2... Step: 32... Loss: 1.061865... Acc: 0.541667
Epoch: 1/2... Step: 33... Loss: 1.074052... Acc: 0.416667
Epoch: 1/2... Step: 34... Loss: 1.095657... Acc: 0.375000
Epoch: 1/2... Step: 35... Loss: 1.073908... Acc: 0.395833
Epoch: 1/2... Step: 36... Loss: 1.112280... Acc: 0.333333
Epoch: 1/2... Step: 37... Loss: 1.046075... Acc: 0.479167
Epoch: 1/2... Step: 38... Loss: 1.107544... Acc: 0.312500
Epoch: 1/2... Step: 39... Loss: 1.054420... Acc: 0.458333
Epoch: 1/2... Step: 40... Loss: 1.078307... Acc: 0.354167
Epoch: 1/2... Step: 41... Loss: 1.065055... Acc: 0.458333
Epoch: 1/2... Step: 42... Loss: 1.076734... Acc: 0.354167
Epoch: 1/2... Step: 43... Loss: 1.103078... Acc: 0.354167
Epoch: 1/2... Step: 44... Loss: 1.094546... Acc: 0.437500
Epoch: 1/2... Step: 45... Loss: 1.027027... Acc: 0.416667
Epoch: 1/2... Step: 46... Loss: 1.077533... Acc: 0.479167
Epoch: 1/2... Step: 47... Loss: 1.120493... Acc: 0.291667
Epoch: 1/2... Step: 48... Loss: 1.034057... Acc: 0.458333
Epoch: 1/2... Step: 49... Loss: 1.060692... Acc: 0.520833
Epoch: 1/2... Step: 50... Loss: 1.101632... Acc: 0.416667
Epoch: 1/2... Step: 51... Loss: 1.055957... Acc: 0.437500
```

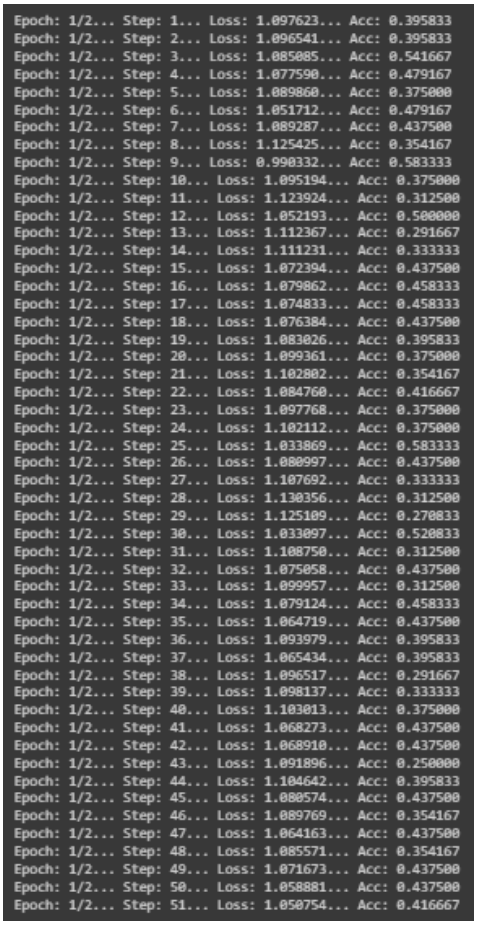
```
Test Loss: 0.670... Test Acc: 42.387%
```

We see a slight difference in results between the two values (around 4%). Unfortunately, this value is not large enough for us to modify our model.

Impact of the `n_layers`

The `n_layers` is ...

The default value is 2, so we also tested 1. Here is the result we obtained:




```
Test Loss: 0.677... Test Acc: 42.059%
```

As above, we can't see any real difference with what we get with the default values.

4. How have you improved the results?

As you can see from the tests above, we couldn't find any specific value in the parameters that would really alter the accuracy of our model, so we decided to keep the initial model.

Results:

Explain the following in this page:

1. Results of training (loss and accuracy of the model)

For the train, we have got a Train Loss of `0.909...` and an accuracy of `63.537%` (with similar value with changes of the random seed). We are far from the 100% success rate, but it is still pretty great, especially considering that our model has to determine between three

different categories (and not only two). (I mean, I'm pretty sure some humans would be under that `63.537%` rate.)

2. Results on testing (loss and accuracy of the model)

For the test, we have got a Test Loss of 0.977 and an accuracy of 61.688% (with similar value with changes of the random seed). It's under the one for the training (of course) but it is still pretty great and pretty close, especially considering that our model has to determine between three different categories (and not only two). It proves that the model could actually be used on non already-processed data and be useful.

3. Show some direct results: example input picture/ sentence and output results.

The last part allow us to do so, here is an example :

```
Input: I love this movie!
Predicted Sentiment: positive
================================================
Input: This is a terrible film.
Predicted Sentiment: positive
================================================
Input: The movie is neither good nor bad.
Predicted Sentiment: positive
================================================
Input: Sooo SAD I will miss you here in San Diego!!!
Predicted Sentiment: negative
================================================
Input: I Love you
Predicted Sentiment: positive
================================================
```