Final Project

NLP Based Projects

Sentiment Analysis : Find the sentiment of a Tweet, whether it is positive or negative.

Link:

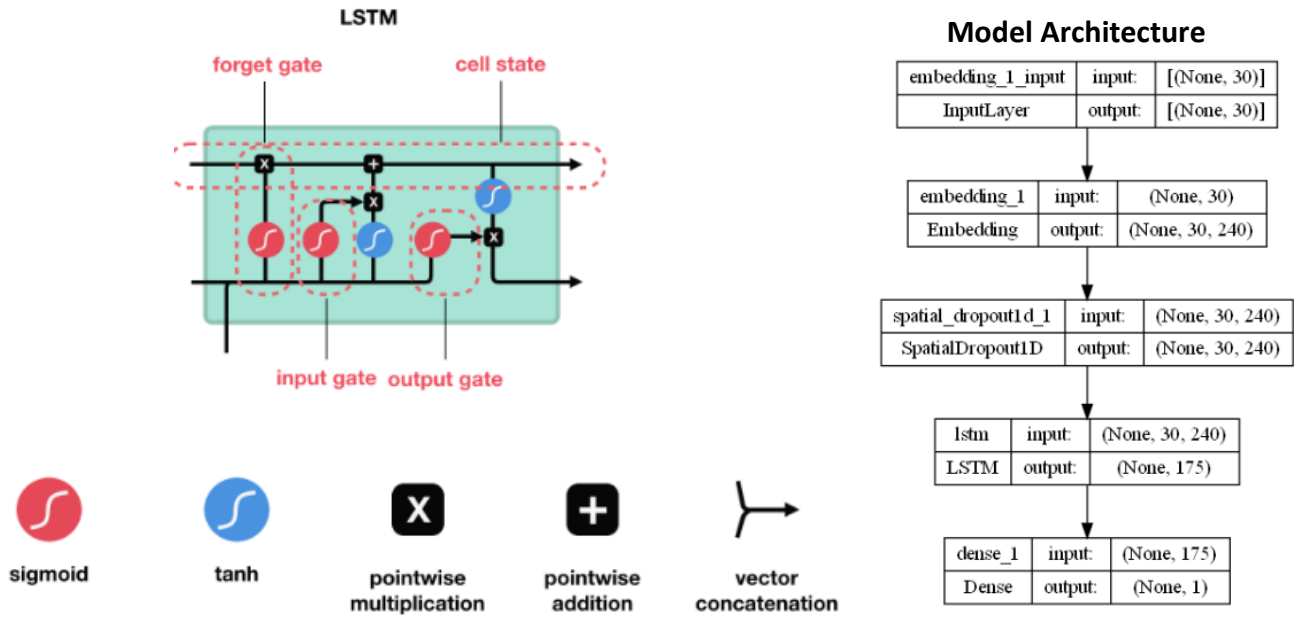
Project link (Github/ Gitlab)

Team Members:

Student Name	Student ID	Contribution in the project
Lucien Claeys	73181	Data cleaning
Theo Chichery	73165	Model development
Martin Joubert	73154	Hyperparameters tuning
Jean-Bastien Morales	73113	Model evaluation

Model Architecture:

The architecture used for this project is an LSTM architecture.



The model have 3 layers like we can see on the right schematic. Not to mention the dense, embedding_1_input layer that represents programme input and output

There is only one hidden layer, the Lstm layer, and it has 175 neurons.

```
model.add(LSTM(lstm_units, dropout=0.5, recurrent_dropout=0.5))
```

```
lstm_units = 175
```

During the project we added layers to improve the accuracy of the model with the embedding and spatial dropout layers.

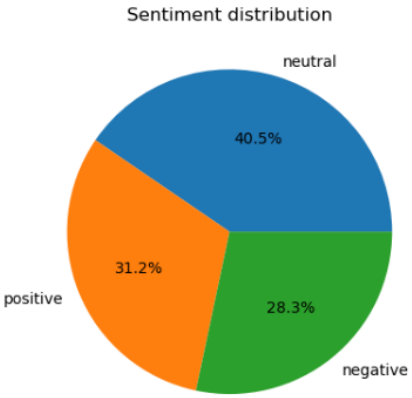
```
model.add(SpatialDropout1D(0.3))
```

```
model.add(Embedding(vocab_size, embed_dim, input_length=X_train.shape[1]))
```

Dataset Description:

We have a dataset with 4 columns (id, text,selected_text and sentiment) of 27480 rows with 3 possible types of sentiment (positive,neutral negative) see the table on the left. The diagram on the right shows a balanced distribution of each sentiment.

	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
...	...	...	...	...
27476	4eac33d1c0	wish we could come see u on Denver husband l...	d lost	negative



To use the data in the model, delete the id and text columns. Then we delete the rows with the neutral sentiments as we want a model that detects whether a tweet is positive or negative. Finally, we remove the empty values and set the negative sentiments to 0 and the positive to 1, giving the final data set :

	selected_text	sentiment
1	sooo sad	0
2	bullying me	0
3	leave me alone	0
4	sons of ****,	0
6	fun	1
9	wow... u just became cooler.	1
11	like	1
12	dangerously	0
13	lost	0
15	uh oh, i am sunburned	0

For the split test train we used the train_test_split function in sklearn.model_selection. We take 80% of the data for the data in the train and therefore 20% for the train, with a mixture of data.

```
train data, test data = train test split(tweet, test size=0.2, random state=16)
```

To increase the size of the dataset, we used EDA, and more specifically the function that replaces a word with a synonym, which multiplies the size of the dataset by 2.

```
augmented_dfs = []
for i in range (tweet.shape[0]):
    augmented_text = eda.synonym_replacement(tweet['selected_text'].iloc[i])
    augmented_df = pd.DataFrame({'selected_text': [augmented_text], 'sentiment': tweet['sentiment'].iloc[i]})
    augmented_dfs.append(augmented_df)
tweet = pd.concat([tweet]+ augmented_dfs, ignore_index=True)
tweet
```

Example

Methodology:

Training parameters

- 'embed_dim': 50
- 'spatial_dropout': 0.4
- 'lstm_units': 150
- 'lstm_dropout': 0.30000000000000004
- 'recurrent_dropout': 0.4
- 'learning_rate': 0.0001

Loss function

loss='binary_crossentropy'

Binary crossentropy is like a scorecard that your model uses to learn and improve its predictions for binary classification problems.

Parameters tuning

- **Epochs** (epochs=5): Number of passes through the entire dataset during training. More epochs may improve performance but can lead to overfitting.
- **Embedding Dimension** (embed_dim=50): Size of word vector space. Higher values capture more complex relationships but increase complexity.
- Spatial Dropout (spatial_dropout=0.4): Dropout on input data, preventing overfitting by introducing noise. Higher values may increase generalization but slow down training.
- LSTM Units (lstm_units=150): Number of memory units in the LSTM layer. More units capture complex patterns but increase model complexity.
- LSTM Dropout (lstm_dropout=0.3): Dropout on connections within LSTM units. Prevents overfitting, with higher values potentially increasing generalization.
- Recurrent Dropout (recurrent_dropout=0.4): Dropout on recurrent connections in LSTM. Prevents overfitting in temporal relationships.
- Learning Rate (learning_rate=0.0001): Step size during optimization. Higher values may speed up training but risk overshooting. Critical for convergence and often requires tuning.

Results improvement

To improve my results, I have used a keras function called keras_tuner that works quite similar to GridSearchCV, testing multiple hyperparameters and find the best one to improve results. On this code, I have set the objective to val_loss trying to reduce it as much as possible.

Results:

Model evaluation

```
Epoch 1/5
131/131 [==============================] - 31s 217ms/step - loss: 0.1457 - accuracy: 0.9554
Epoch 2/5
131/131 [==============================] - 28s 217ms/step - loss: 0.1255 - accuracy: 0.9597
Epoch 3/5
131/131 [==============================] - 31s 238ms/step - loss: 0.1147 - accuracy: 0.9647
Epoch 4/5
131/131 [==============================] - 29s 219ms/step - loss: 0.1070 - accuracy: 0.9671
Epoch 5/5
131/131 [==============================] - 29s 223ms/step - loss: 0.1038 - accuracy: 0.9672
```

Here are the results of our best model fitting. As we can see both loss and accuracy are very good, resulting from 6 trials with different parameters using keras tuner.

```
103/103 [==============================] - 2s 22ms/step - loss: 0.3046 - accuracy: 0.8921
Test Loss: 0.304638
Test Accuracy: 0.892148
```

But after evaluating our model with the testing data, our results lower significantly getting 30% of losses and 89% of accuracy. However, it is quite normal and is called overfitting, occurring when a model, having memorized the training data, fails to generalize well to new, unseen data, leading to lower accuracy and higher losses on the test set.

Applying the code to new data

We can see that the code is good at identifying negative and positive feelings, but sometimes misidentifies neutral feelings.

```
1/1 [==============================] - 0s 40ms/step
Phrase: My name is Theo
Sentiment prédit: negative
Score de confiance: 0.029197754338383675
```

```
Phrase: I love
Sentiment prédit: positive
Score de confiance: 0.9980809688568115
```

```
1/1 [==============================] - 0s 47ms/step
Phrase: I'm lost
Sentiment prédit: negative
Score de confiance: 0.0005198196158744395
```

```
1/1 [==============================] - 0s 31ms/step
Phrase: I am new
Sentiment prédit: neutral
Score de confiance: 0.8811243176460266
```

```
1/1 [==============================] - 0s 54ms/step
Phrase: I find you
Sentiment prédit: neutral
Score de confiance: 0.12290730327367783
```