NLP TP5 REPORT

Machine Translation
English to French

2024

- SEHILI CHAIMA
- HACHOUD MOHAMMED

GROUP 01

INTRODUCTION

Machine translation (MT) refers to fully automated software that can translate source content into target languages. Humans may use MT to help them render text and speech into another language, or the MT software may operate without human intervention.

We will build Neural network architecture (GRU) to train the algorithm translate from English to French. We will use keras library to build Neural Network and then obtain the translation.

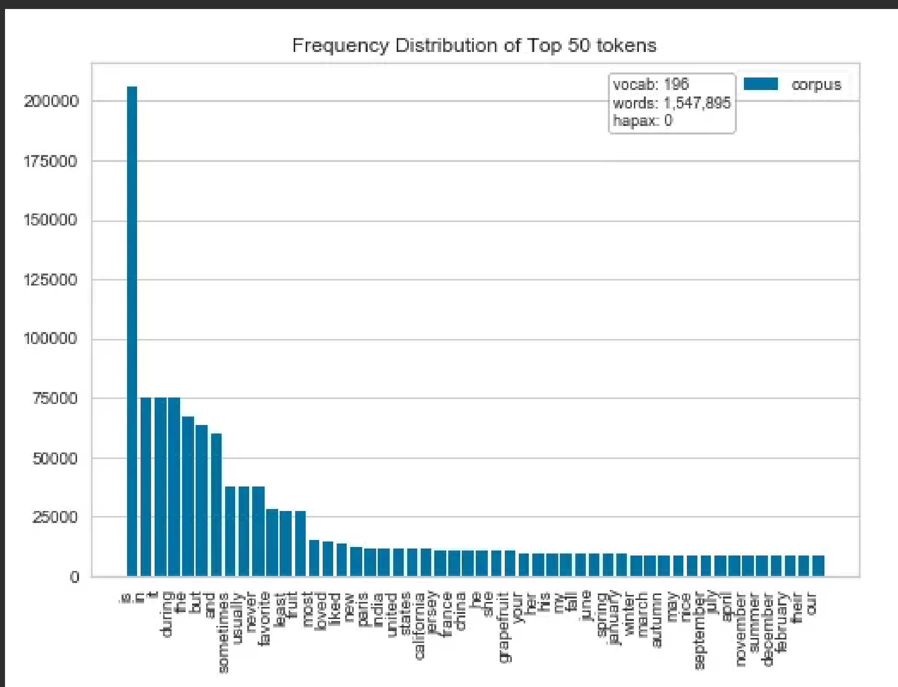
DATASET

The most common datasets used for machine translation are from WMT. However, that will take a long time to train a neural network on. We'll be using a dataset we created for this project that contains a small vocabulary. We'll be able to train your model in a reasonable time with this dataset.

Overview of our dataset

```
small_vocab_en Line 1:  new jersey is sometimes quiet during autumn , and it is snowy in april .
small_vocab_fr Line 1:  new jersey est parfois calme pendant l' automne , et il est neigeux en avril .
small_vocab_en Line 2:  the united states is usually chilly during july , and it is usually freezing in november .
small_vocab_fr Line 2:  les Ã©tats-unis est gÃ©nÃ©ralement froid en juillet , et il gÃ¨le habituellement en novembre .
```

- The `small_vocab_en` file contains English sentences with their French translations in the `small_vocab_fr` file.



Vocabulary

- our dataset has **1823250 English words** with **227 unique words** and **1961295 French words with 355 unique words.**

Data Preprocessing

Tokenization

- map words to ids in order to make them comprehensible to machine learning algorithms.

```
Sequence 1 in x
  Input:  The quick brown fox jumps over the lazy dog .
  Output: [1, 2, 4, 5, 6, 7, 1, 8, 9]
Sequence 2 in x
  Input:  By Jove , my quick study of lexicography won a prize .
  Output: [10, 11, 12, 2, 13, 14, 15, 16, 3, 17]
Sequence 3 in x
  Input:  This is a short sentence .
  Output: [18, 19, 3, 20, 21]
```

Padding

- Make sure all the English and french sequences have the same length.

```
Sequence 1 in x
  Input:  [1 2 4 5 6 7 1 8 9]
  Output: [1 2 4 5 6 7 1 8 9 0]
Sequence 2 in x
  Input:  [10 11 12  2 13 14 15 16  3 17]
  Output: [10 11 12  2 13 14 15 16  3 17]
Sequence 3 in x
  Input:  [18 19  3 20 21]
  Output: [18 19  3 20 21  0  0  0  0  0]
```

Preprocessed Data

```
Data Preprocessed
Max English sentence length: 15
Max French sentence length: 21
English vocabulary size: 199
French vocabulary size: 345
```

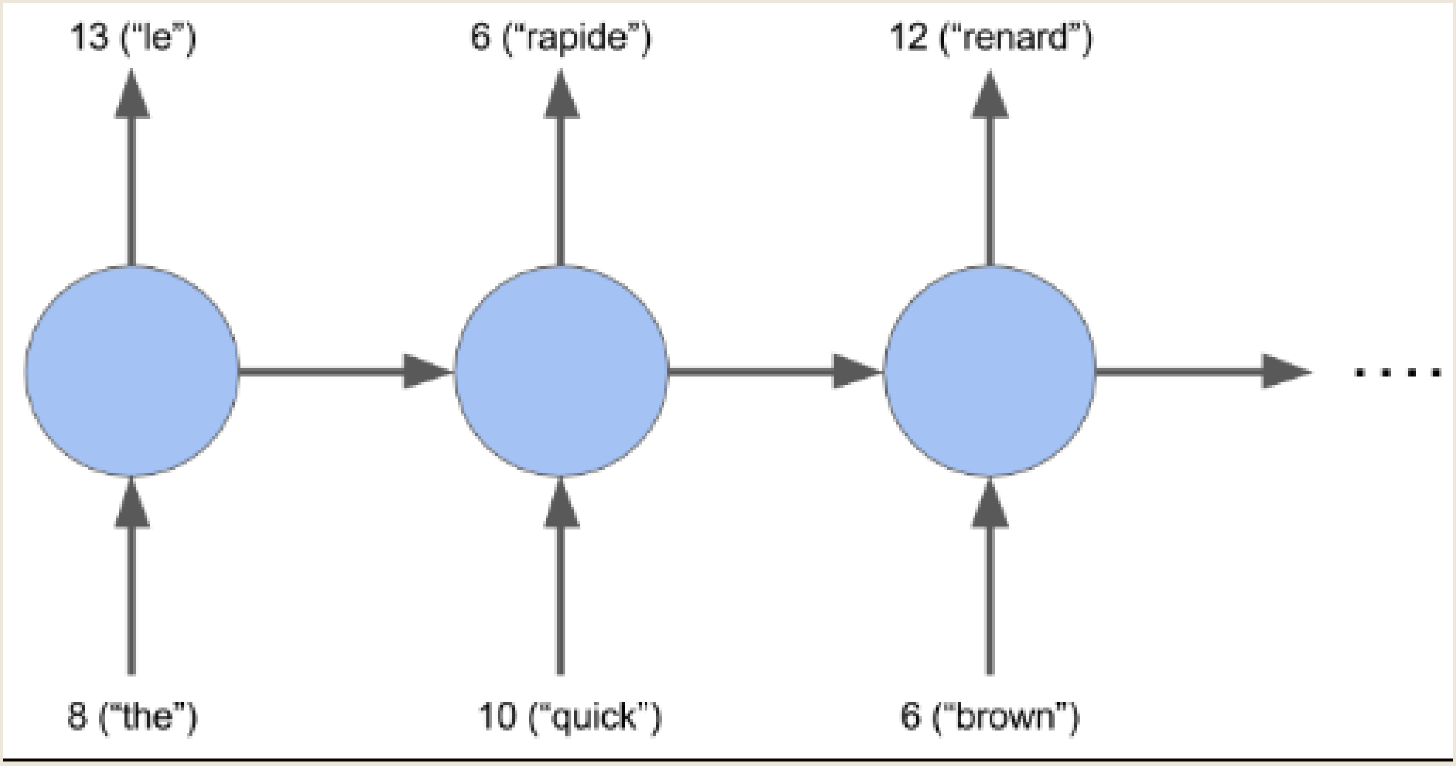
Models Training, Evaluation and Validation

- We have implemented **three model** architectures.
- Models are evaluated by the **categorical cross entropy** on validation data.

Function to map ids back to words.

```
def logits_to_text(logits, tokenizer):

    index_to_words = {id: word for word, id in tokenizer.word_index.items()}
    index_to_words[0] = '<PAD>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1) if index_to_words[prediction]!='<PAD>'] )

print('`logits_to_text` function loaded.')
```

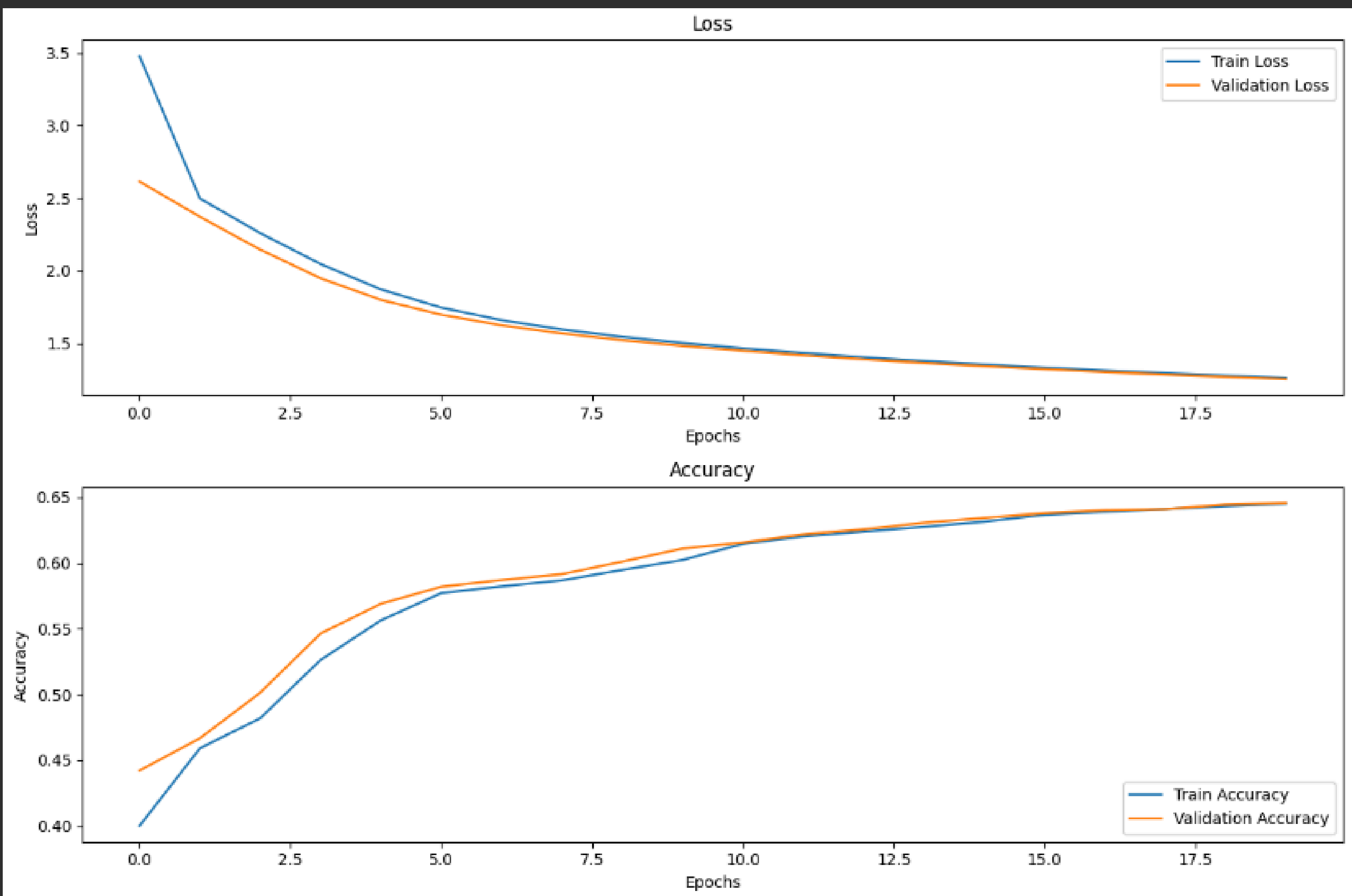
Model 1: Basic RNN with Time Distributed Layer



```
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 21, 1)]           0

 LSTM_layer (LSTM)           (None, 21, 64)            16896

 Dense_layer (TimeDistribut  (None, 21, 345)           22425
 ed)

=================================================================
Total params: 39321 (153.60 KB)
Trainable params: 39321 (153.60 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
Epoch 1/20
108/108 [==============================] - 41s 355ms/step - loss: 3.4514 - accuracy: 0.4053 - val_loss: 2.5771 - val_accuracy: 0.4635
Epoch 2/20
108/108 [==============================] - 41s 384ms/step - loss: 2.4480 - accuracy: 0.4707 - val_loss: 2.3122 - val_accuracy: 0.4866
Epoch 3/20
108/108 [==============================] - 37s 335ms/step - loss: 2.1907 - accuracy: 0.4995 - val_loss: 2.0676 - val_accuracy: 0.5252
Epoch 4/20
108/108 [==============================] - 35s 328ms/step - loss: 1.9681 - accuracy: 0.5465 - val_loss: 1.8756 - val_accuracy: 0.5633
Epoch 5/20
...
Epoch 19/20
108/108 [==============================] - 33s 308ms/step - loss: 1.2883 - accuracy: 0.6437 - val_loss: 1.2794 - val_accuracy: 0.6454
Epoch 20/20
108/108 [==============================] - 34s 313ms/step - loss: 1.2738 - accuracy: 0.6452 - val_loss: 1.2655 - val_accuracy: 0.6462
```

Learning curves:



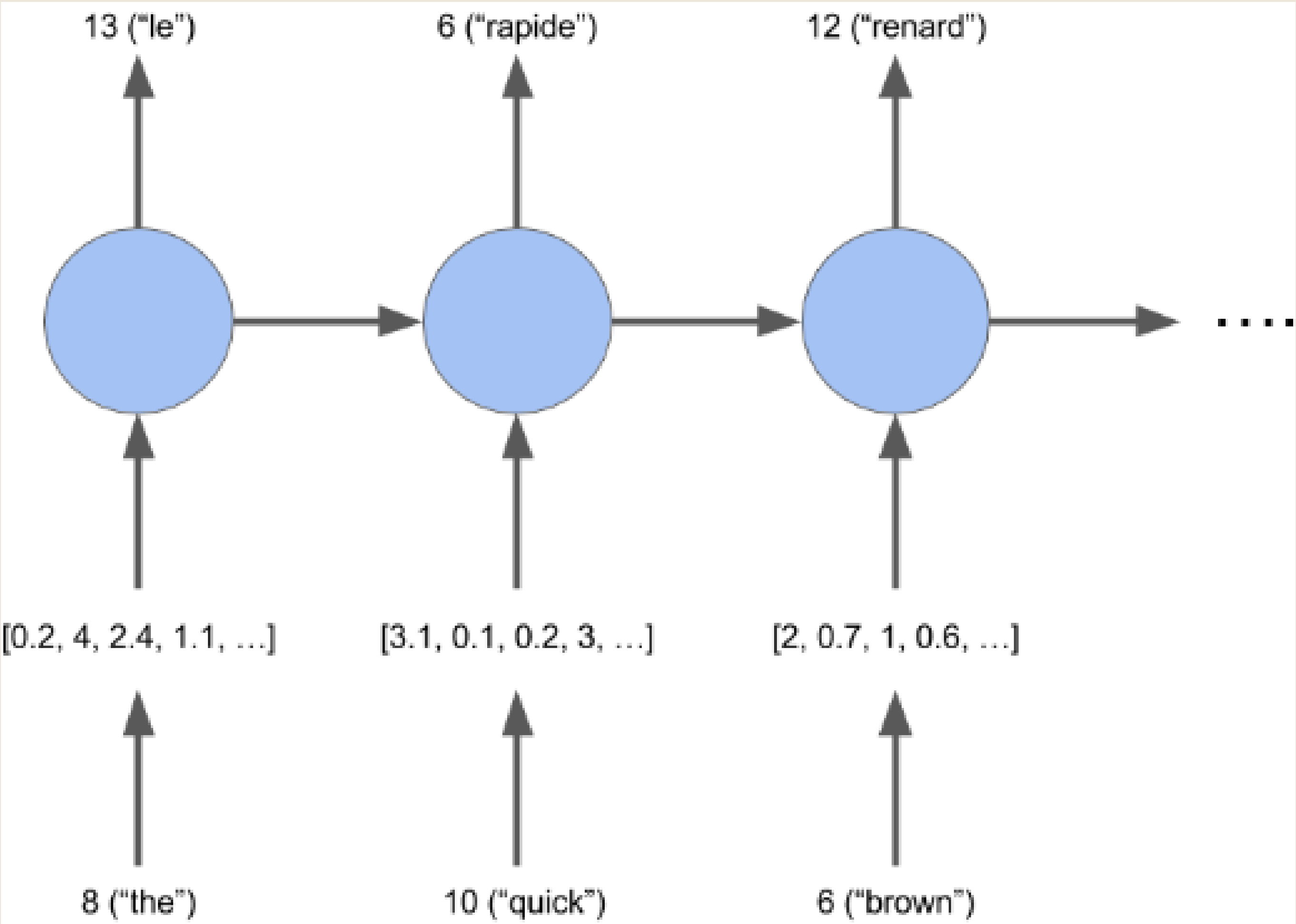
- Accuracy and loss are converging properly, we have achieved a validation accuracy of 64%, we will try other improved architectures and maybe training on more epochs would be also useful
- when it comes to the variance between validation and training sets, it’s approximately not at all variance

Prediction:

```
---- Original ----
new jersey is sometimes quiet during autumn and it is snowy in april

1/1 [==============================] - 1s 606ms/step
---- Gold standard ----
new jersey est parfois calme pendant l' automne et il est neigeux en avril

---- Prediction ----
 new jersey est parfois calme en l' et il est est en en
```

```
---- Original ----
his favorite fruit is the orange but my favorite is the grape

1/1 [==============================] - 0s 115ms/step
---- Gold standard ----
son fruit préféré est l'orange mais mon préféré est le raisin

---- Prediction ----
 elle fruit préféré est la mais mais son préféré est la
```

Model 2: Basic RNN with Time Distributed Layer and Embedding

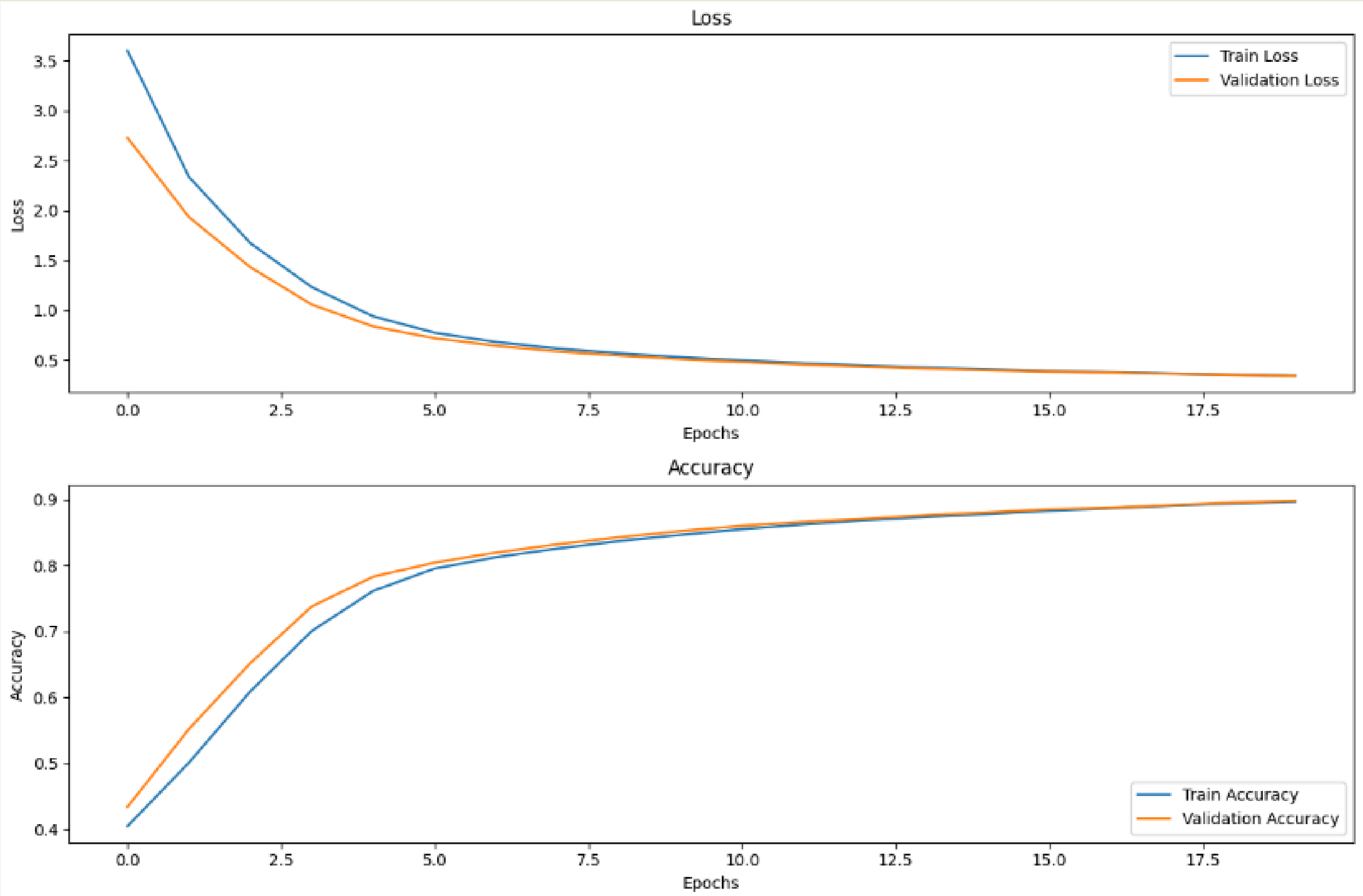



```
Model: "Embedding_LSTM"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 21)]              0

 Embedding_layer (Embedding  (None, 21, 256)           51200
 )

 LSTM_layer (LSTM)           (None, 21, 64)            82176

 Dense_layer (TimeDistribut  (None, 21, 345)           22425
 ed)

=================================================================
Total params: 155801 (608.60 KB)
Trainable params: 155801 (608.60 KB)
Non-trainable params: 0 (0.00 Byte)
```

```
Epoch 1/20
108/108 [==============================] - 63s 552ms/step - loss: 3.5993 - accuracy: 0.4049 - val_loss: 2.7268 - val_accuracy: 0.4342
Epoch 2/20
108/108 [==============================] - 56s 516ms/step - loss: 2.3342 - accuracy: 0.5012 - val_loss: 1.9324 - val_accuracy: 0.5521
Epoch 3/20
108/108 [==============================] - 55s 509ms/step - loss: 1.6702 - accuracy: 0.6093 - val_loss: 1.4291 - val_accuracy: 0.6520
...
Epoch 19/20
108/108 [==============================] - 64s 591ms/step - loss: 0.3531 - accuracy: 0.8936 - val_loss: 0.3500 - val_accuracy: 0.8950
Epoch 20/20
108/108 [==============================] - 60s 560ms/step - loss: 0.3433 - accuracy: 0.8963 - val_loss: 0.3399 - val_accuracy: 0.8974
```

- Validation Accuracy is now **84%**, we notice that this is a huge improvement after adding just the **embedding layer**.

Learning curves:



Prediction:

```
---- Original ----
new jersey is sometimes quiet during autumn and it is snowy in april

1/1 [==============================] - 1s 923ms/step
---- Gold standard ----
new jersey est parfois calme pendant l' automne et il est neigeux en avril

---- Prediction ----
 new jersey est parfois calme en l' et il automne est est en avril
```

- Better prediction compared to the first model.

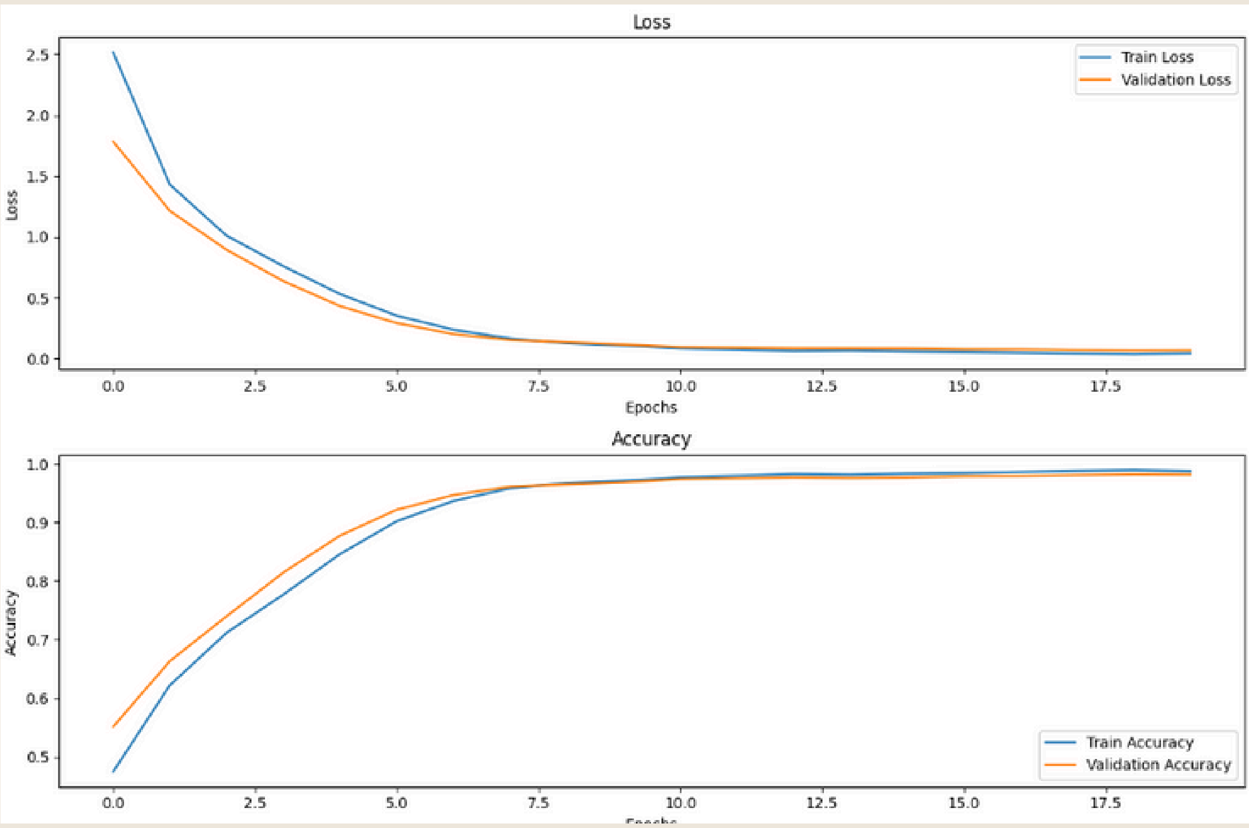
Model 3: bi-LSTM encoder LSTM decoder

- We tried to improve more in the model architecture and this encoder decoder based architecture gave us the best results : **validation accuracy of 98% after 20 epochs, although the training time was 3 hours.**

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Embedding_layer (Embedding  (None, 21, 256)           51200
 )

 Bi_LSTM_encoder (Bidirecti  (None, 512)               1050624
 onal)

 Glue (RepeatVector)         (None, 21, 512)           0

 LSTM_decoder (LSTM)         (None, 21, 256)           787456

 Dense (TimeDistributed)     (None, 21, 345)           88665

=================================================================
```

```
Epoch 1/20
108/108 [==============================] - 456s 4s/step - loss: 2.5165 - accuracy: 0.4745 - val_loss: 1.7836 - val_accuracy: 0.5513
Epoch 2/20
108/108 [==============================] - 437s 4s/step - loss: 1.4304 - accuracy: 0.6223 - val_loss: 1.2145 - val_accuracy: 0.6635
...
108/108 [==============================] - 428s 4s/step - loss: 0.0371 - accuracy: 0.9899 - val_loss: 0.0665 - val_accuracy: 0.9826
Epoch 20/20
108/108 [==============================] - 433s 4s/step - loss: 0.0437 - accuracy: 0.9879 - val_loss: 0.0683 - val_accuracy: 0.9821
Final Model Loaded
```

Learning curves:



Prediction:

```
---- Original ----
new jersey is sometimes quiet during autumn and it is snowy in april

1/1 [==============================] - 2s 2s/step
---- Gold standard ----
new jersey est parfois calme pendant l' automne et il est neigeux en avril

---- Prediction ----
 new jersey est parfois calme pendant l' automne et il est neigeux en avril
```

- It has predicted correctly the whole sentence comparing to the previous models that mispredicted some words.