In [1]:

```
import collections

import helper
import numpy as np


from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model
from keras.layers import GRU, Input, Dense, TimeDistributed, Activation, RepeatVector, B
idirectional
from keras.layers.embeddings import Embedding
from keras.optimizers import Adam
from keras.losses import sparse_categorical_crossentropy
```

In [3]:

```
# mount the google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:

```
filedir ="/content/drive/MyDrive/data"
```

In [6]:

```
import os
files = os.listdir(filedir)
print(files)
```

```
['small_vocab_fr.csv', 'small_vocab_en.csv', 'final_model', '.ipynb_checkpoints']
```

In [7]:

```
import os


def load_data(path):
    """
    Load dataset
    """
    input_file = os.path.join(path)
    with open(input_file, "r") as f:
        data = f.read()

    return data.split('\n')
f1 = files[0]
f2 = files[1]
english_sentences = load_data(r'/content/drive/MyDrive/data/small_vocab_en.csv')
french_sentences = load_data(r'/content/drive/MyDrive/data/small_vocab_fr.csv')
```

In [8]:

```
english_words_counter = collections.Counter([word for sentence in english_sentences for
word in sentence.split()])
french_words_counter = collections.Counter([word for sentence in french_sentences for wor
d in sentence.split()])

print('{} English words.'.format(len([word for sentence in english_sentences for word in
sentence.split()])))
print('{} unique English words.'.format(len(english_words_counter)))
print('10 Most common words in the English dataset:')
print('"' + '" "'.join(list(zip(*english_words_counter.most_common(10)))[0]) + '"')
```

```
print()
print('{} French words.'.format(len([word for sentence in french_sentences for word in se
ntence.split()])))
print('{} unique French words.'.format(len(french_words_counter)))
print('10 Most common words in the French dataset:')
print('"' + '" "'.join(list(zip(*french_words_counter.most_common(10)))[0]) + '"')
```

```
1823250 English words.
227 unique English words.
10 Most common words in the English dataset:
"is" "," "." "in" "it" "during" "the" "but" "and" "sometimes"

1961295 French words.
355 unique French words.
10 Most common words in the French dataset:
"est" "." "," "en" "il" "les" "mais" "et" "la" "parfois"
```

In [9]:

```
def tokenize(x):
    """
    Tokenize x
    :param x: List of sentences/strings to be tokenized
    :return: Tuple of (tokenized x data, tokenizer used to tokenize x)
    """
    tokenizer=Tokenizer()
    tokenizer.fit_on_texts(x)
    t=tokenizer.texts_to_sequences(x)
    # TODO: Implement
    return t, tokenizer


# Tokenize Example output
text_sentences = [
    'The quick brown fox jumps over the lazy dog .',
    'By Jove , my quick study of lexicography won a prize .',
    'This is a short sentence .']
text_tokenized, text_tokenizer = tokenize(text_sentences)
print(text_tokenizer.word_index)
print()
for sample_i, (sent, token_sent) in enumerate(zip(text_sentences, text_tokenized)):
    print('Sequence {} in x'.format(sample_i + 1))
    print('  Input:  {}'.format(sent))
    print('  Output: {}'.format(token_sent))
```

```
{'the': 1, 'quick': 2, 'a': 3, 'brown': 4, 'fox': 5, 'jumps': 6, 'over': 7, 'lazy': 8, 'do
g': 9, 'by': 10, 'jove': 11, 'my': 12, 'study': 13, 'of': 14, 'lexicography': 15, 'won':
16, 'prize': 17, 'this': 18, 'is': 19, 'short': 20, 'sentence': 21}

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]
```

In [10]:

```
def pad(x, length=None):
    """
    Pad x
    :param x: List of sequences.
    :param length: Length to pad the sequence to.  If None, use length of longest sequenc
e in x.
    :return: Padded numpy array of sequences
    """
    # TODO: Implement
    padding=pad_sequences(x,padding='post',maxlen=length)
    return padding
```

```
# Pad Tokenized output
test_pad = pad(text_tokenized)
for sample_i, (token_sent, pad_sent) in enumerate(zip(text_tokenized, test_pad)):
    print('Sequence {} in x'.format(sample_i + 1))
    print('  Input:  {}'.format(np.array(token_sent)))
    print('  Output: {}'.format(pad_sent))
```

```
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]
```

In [11]:

```
def preprocess(x, y):
    """
    Preprocess x and y
    :param x: Feature List of sentences
    :param y: Label List of sentences
    :return: Tuple of (Preprocessed x, Preprocessed y, x tokenizer, y tokenizer)
    """
    preprocess_x, x_tk = tokenize(x)
    preprocess_y, y_tk = tokenize(y)

    preprocess_x = pad(preprocess_x)
    preprocess_y = pad(preprocess_y)

    # Keras's sparse_categorical_crossentropy function requires the labels to be in 3 dim
ensions
    preprocess_y = preprocess_y.reshape(*preprocess_y.shape, 1)

    return preprocess_x, preprocess_y, x_tk, y_tk

preproc_english_sentences, preproc_french_sentences, english_tokenizer, french_tokenizer
=\
    preprocess(english_sentences, french_sentences)

max_english_sequence_length = preproc_english_sentences.shape[1]
max_french_sequence_length = preproc_french_sentences.shape[1]
english_vocab_size = len(english_tokenizer.word_index)
french_vocab_size = len(french_tokenizer.word_index)

print('Data Preprocessed')
print("Max English sentence length:", max_english_sequence_length)
print("Max French sentence length:", max_french_sequence_length)
print("English vocabulary size:", english_vocab_size)
print("French vocabulary size:", french_vocab_size)
```

```
Data Preprocessed
Max English sentence length: 15
Max French sentence length: 21
English vocabulary size: 199
French vocabulary size: 344
```

In [12]:

```
print(preproc_english_sentences[:1])
```

```
[[17 23  1  8 67  4 39  7  3  1 55  2 44  0  0]]
```

In [13]:

```
def logits_to_text(logits, tokenizer):
    """
    Turn logits from a neural network into text using the tokenizer
```

```
    :param logits: Logits from a neural network
    :param tokenizer: Keras Tokenizer fit on the labels
    :return: String that represents the text of the logits
    """
    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)])
```

In [14]:

```
from keras.layers import Input
from keras.models import Sequential
```

In [15]:

```
def simple_model(input_shape, output_sequence_length, english_vocab_size, french_vocab_si
ze):
    """
    Build and train a basic RNN on x and y
    :param input_shape: Tuple of input shape
    :param output_sequence_length: Length of output sequence
    :param english_vocab_size: Number of unique English words in the dataset
    :param french_vocab_size: Number of unique French words in the dataset
    :return: Keras model built, but not trained
    """
    # TODO: Build the layers
    learning_rate = 1e-3
    input_seq = Input(input_shape[1:])
    rnn = GRU(64, return_sequences = True)(input_seq)
    logits = TimeDistributed(Dense(french_vocab_size+1))(rnn)
    model = Model(input_seq, Activation('softmax')(logits))
    model.compile(loss = sparse_categorical_crossentropy,
                  optimizer = Adam(learning_rate),
                  metrics = ['accuracy'])

    return model


# Reshaping the input to work with a basic RNN
tmp_x = pad(preproc_english_sentences, max_french_sequence_length)
tmp_x = tmp_x.reshape((-1, preproc_french_sentences.shape[-2], 1))

# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024, epochs=10, valida
tion_split=0.2)

# Print prediction(s)
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))
```

```
Epoch 1/10
108/108 [==============================] - 53s 303ms/step - loss: 4.3759 - accuracy: 0.37
14 - val_loss: 2.5872 - val_accuracy: 0.4586
Epoch 2/10
108/108 [==============================] - 31s 291ms/step - loss: 2.5301 - accuracy: 0.46
24 - val_loss: 2.3686 - val_accuracy: 0.4800
Epoch 3/10
108/108 [==============================] - 32s 297ms/step - loss: 2.3142 - accuracy: 0.48
99 - val_loss: 2.1412 - val_accuracy: 0.5268
Epoch 4/10
108/108 [==============================] - 34s 313ms/step - loss: 2.0791 - accuracy: 0.53
09 - val_loss: 1.8976 - val_accuracy: 0.5624
Epoch 5/10
108/108 [==============================] - 32s 292ms/step - loss: 1.8528 - accuracy: 0.56
80 - val_loss: 1.7534 - val_accuracy: 0.5749
Epoch 6/10
108/108 [==============================] - 32s 294ms/step - loss: 1.7302 - accuracy: 0.57
```

```
75 - val_loss: 1.6649 - val_accuracy: 0.5821
Epoch 7/10
108/108 [==============================] - 32s 295ms/step - loss: 1.6491 - accuracy: 0.58
21 - val_loss: 1.5985 - val_accuracy: 0.5842
Epoch 8/10
108/108 [==============================] - 32s 295ms/step - loss: 1.5829 - accuracy: 0.58
73 - val_loss: 1.5407 - val_accuracy: 0.5922
Epoch 9/10
108/108 [==============================] - 32s 297ms/step - loss: 1.5282 - accuracy: 0.59
50 - val_loss: 1.4914 - val_accuracy: 0.6024
Epoch 10/10
108/108 [==============================] - 32s 294ms/step - loss: 1.4794 - accuracy: 0.60
45 - val_loss: 1.4506 - val_accuracy: 0.6090
new jersey est parfois parfois en l' mais il est est en en <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD>
```

In [ ]:

```
# Print prediction(s)
print("Prediction:")
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))

print("\nCorrect Translation:")
print(french_sentences[:1])

print("\nOriginal text:")
print(english_sentences[:1])
```

```
Prediction:
new jersey est parfois parfois en en et il est est en en <PAD> <PAD> <PAD> <PAD> <PAD> <P
AD> <PAD> <PAD>

Correct Translation:
["new jersey est parfois calme pendant l' automne , et il est neigeux en avril ."]

Original text:
['new jersey is sometimes quiet during autumn , and it is snowy in april .']
```

In [ ]:

```
#changing no of epochs from 10 to 20
# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024, epochs=20, valida
tion_split=0.2)
```

```
Epoch 1/20
108/108 [==============================] - 32s 283ms/step - loss: 4.5380 - accuracy: 0.36
51 - val_loss: 2.5692 - val_accuracy: 0.4584
Epoch 2/20
108/108 [==============================] - 30s 279ms/step - loss: 2.5091 - accuracy: 0.46
20 - val_loss: 2.3129 - val_accuracy: 0.4838
Epoch 3/20
108/108 [==============================] - 30s 281ms/step - loss: 2.2335 - accuracy: 0.50
42 - val_loss: 1.9947 - val_accuracy: 0.5556
Epoch 4/20
108/108 [==============================] - 30s 281ms/step - loss: 1.9387 - accuracy: 0.55
99 - val_loss: 1.8002 - val_accuracy: 0.5759
Epoch 5/20
108/108 [==============================] - 30s 282ms/step - loss: 1.7684 - accuracy: 0.57
77 - val_loss: 1.6881 - val_accuracy: 0.5835
Epoch 6/20
108/108 [==============================] - 30s 280ms/step - loss: 1.6691 - accuracy: 0.58
29 - val_loss: 1.6155 - val_accuracy: 0.5899
Epoch 7/20
108/108 [==============================] - 30s 281ms/step - loss: 1.6032 - accuracy: 0.59
04 - val_loss: 1.5577 - val_accuracy: 0.5969
Epoch 8/20
```

```
108/108 [==============================] - 30s 280ms/step - loss: 1.5439 - accuracy: 0.59
82 - val_loss: 1.5063 - val_accuracy: 0.6100
Epoch 9/20
108/108 [==============================] - 30s 282ms/step - loss: 1.4951 - accuracy: 0.60
74 - val_loss: 1.4586 - val_accuracy: 0.6113
Epoch 10/20
108/108 [==============================] - 30s 280ms/step - loss: 1.4457 - accuracy: 0.61
42 - val_loss: 1.4153 - val_accuracy: 0.6252
Epoch 11/20
108/108 [==============================] - 30s 279ms/step - loss: 1.4047 - accuracy: 0.62
41 - val_loss: 1.3778 - val_accuracy: 0.6277
Epoch 12/20
108/108 [==============================] - 30s 279ms/step - loss: 1.3690 - accuracy: 0.62
92 - val_loss: 1.3459 - val_accuracy: 0.6335
Epoch 13/20
108/108 [==============================] - 30s 279ms/step - loss: 1.3382 - accuracy: 0.63
56 - val_loss: 1.3168 - val_accuracy: 0.6393
Epoch 14/20
108/108 [==============================] - 30s 278ms/step - loss: 1.3128 - accuracy: 0.63
98 - val_loss: 1.2923 - val_accuracy: 0.6411
Epoch 15/20
108/108 [==============================] - 30s 283ms/step - loss: 1.2873 - accuracy: 0.64
35 - val_loss: 1.2706 - val_accuracy: 0.6441
Epoch 16/20
108/108 [==============================] - 30s 281ms/step - loss: 1.2651 - accuracy: 0.64
60 - val_loss: 1.2503 - val_accuracy: 0.6464
Epoch 17/20
108/108 [==============================] - 30s 283ms/step - loss: 1.2450 - accuracy: 0.64
81 - val_loss: 1.2316 - val_accuracy: 0.6485
Epoch 18/20
108/108 [==============================] - 30s 282ms/step - loss: 1.2247 - accuracy: 0.65
12 - val_loss: 1.2167 - val_accuracy: 0.6497
Epoch 19/20
108/108 [==============================] - 30s 280ms/step - loss: 1.2125 - accuracy: 0.65
14 - val_loss: 1.2025 - val_accuracy: 0.6520
Epoch 20/20
108/108 [==============================] - 30s 279ms/step - loss: 1.1980 - accuracy: 0.65
40 - val_loss: 1.1885 - val_accuracy: 0.6539
```

Out[ ]:

```
<keras.callbacks.History at 0x7f956dd68510>
```

In [ ]:

```
# Print prediction(s)
print("Prediction:")
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))

print("\nCorrect Translation:")
print(french_sentences[:1])

print("\nOriginal text:")
print(english_sentences[:1])
```

```
Prediction:
new jersey est parfois calme en l' et il est est en en <PAD> <PAD> <PAD> <PAD> <PAD> <PAD
> <PAD> <PAD>

Correct Translation:
["new jersey est parfois calme pendant l' automne , et il est neigeux en avril ."]

Original text:
['new jersey is sometimes quiet during autumn , and it is snowy in april .']
```

In [ ]:

```
#changing the number of epochs from 20 to 30
# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
```

```
    english_vocab_size,
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024, epochs=30, valida
tion_split=0.2)
```

```
Epoch 1/30
108/108 [==============================] - 33s 283ms/step - loss: 4.4119 - accuracy: 0.36
07 - val_loss: 2.5593 - val_accuracy: 0.4590
Epoch 2/30
108/108 [==============================] - 30s 282ms/step - loss: 2.4940 - accuracy: 0.46
50 - val_loss: 2.3032 - val_accuracy: 0.4777
Epoch 3/30
108/108 [==============================] - 30s 282ms/step - loss: 2.2482 - accuracy: 0.49
65 - val_loss: 2.0799 - val_accuracy: 0.5155
Epoch 4/30
108/108 [==============================] - 30s 282ms/step - loss: 2.0268 - accuracy: 0.53
53 - val_loss: 1.8775 - val_accuracy: 0.5561
Epoch 5/30
108/108 [==============================] - 30s 282ms/step - loss: 1.8350 - accuracy: 0.56
59 - val_loss: 1.7231 - val_accuracy: 0.5851
Epoch 6/30
108/108 [==============================] - 30s 281ms/step - loss: 1.6932 - accuracy: 0.58
49 - val_loss: 1.6250 - val_accuracy: 0.5913
Epoch 7/30
108/108 [==============================] - 30s 282ms/step - loss: 1.6092 - accuracy: 0.59
33 - val_loss: 1.5587 - val_accuracy: 0.6006
Epoch 8/30
108/108 [==============================] - 30s 282ms/step - loss: 1.5446 - accuracy: 0.60
62 - val_loss: 1.5029 - val_accuracy: 0.6134
Epoch 9/30
108/108 [==============================] - 30s 280ms/step - loss: 1.4894 - accuracy: 0.61
62 - val_loss: 1.4534 - val_accuracy: 0.6225
Epoch 10/30
108/108 [==============================] - 30s 282ms/step - loss: 1.4412 - accuracy: 0.62
30 - val_loss: 1.4122 - val_accuracy: 0.6292
Epoch 11/30
108/108 [==============================] - 31s 283ms/step - loss: 1.4028 - accuracy: 0.63
01 - val_loss: 1.3753 - val_accuracy: 0.6320
Epoch 12/30
108/108 [==============================] - 30s 280ms/step - loss: 1.3687 - accuracy: 0.63
59 - val_loss: 1.3423 - val_accuracy: 0.6397
Epoch 13/30
108/108 [==============================] - 30s 278ms/step - loss: 1.3353 - accuracy: 0.64
04 - val_loss: 1.3145 - val_accuracy: 0.6458
Epoch 14/30
108/108 [==============================] - 31s 290ms/step - loss: 1.3087 - accuracy: 0.64
49 - val_loss: 1.2870 - val_accuracy: 0.6442
Epoch 15/30
108/108 [==============================] - 31s 288ms/step - loss: 1.2845 - accuracy: 0.64
59 - val_loss: 1.2638 - val_accuracy: 0.6484
Epoch 16/30
108/108 [==============================] - 31s 291ms/step - loss: 1.2602 - accuracy: 0.64
92 - val_loss: 1.2431 - val_accuracy: 0.6496
Epoch 17/30
108/108 [==============================] - 31s 285ms/step - loss: 1.2383 - accuracy: 0.65
01 - val_loss: 1.2254 - val_accuracy: 0.6521
Epoch 18/30
108/108 [==============================] - 31s 285ms/step - loss: 1.2192 - accuracy: 0.65
32 - val_loss: 1.2091 - val_accuracy: 0.6528
Epoch 19/30
108/108 [==============================] - 31s 288ms/step - loss: 1.2089 - accuracy: 0.65
30 - val_loss: 1.1945 - val_accuracy: 0.6551
Epoch 20/30
108/108 [==============================] - 31s 287ms/step - loss: 1.1894 - accuracy: 0.65
60 - val_loss: 1.1815 - val_accuracy: 0.6562
Epoch 21/30
108/108 [==============================] - 31s 287ms/step - loss: 1.1761 - accuracy: 0.65
69 - val_loss: 1.1701 - val_accuracy: 0.6611
Epoch 22/30
108/108 [==============================] - 31s 288ms/step - loss: 1.1680 - accuracy: 0.65
83 - val_loss: 1.1593 - val_accuracy: 0.6586
Epoch 23/30
```

```
108/108 [==============================] - 31s 286ms/step - loss: 1.1560 - accuracy: 0.65
98 - val_loss: 1.1493 - val_accuracy: 0.6631
Epoch 24/30
108/108 [==============================] - 31s 291ms/step - loss: 1.1467 - accuracy: 0.66
13 - val_loss: 1.1414 - val_accuracy: 0.6608
Epoch 25/30
108/108 [==============================] - 31s 289ms/step - loss: 1.1385 - accuracy: 0.66
13 - val_loss: 1.1312 - val_accuracy: 0.6616
Epoch 26/30
108/108 [==============================] - 31s 286ms/step - loss: 1.1295 - accuracy: 0.66
24 - val_loss: 1.1236 - val_accuracy: 0.6615
Epoch 27/30
108/108 [==============================] - 31s 286ms/step - loss: 1.1224 - accuracy: 0.66
29 - val_loss: 1.1150 - val_accuracy: 0.6651
Epoch 28/30
108/108 [==============================] - 31s 288ms/step - loss: 1.1162 - accuracy: 0.66
46 - val_loss: 1.1085 - val_accuracy: 0.6620
Epoch 29/30
108/108 [==============================] - 31s 291ms/step - loss: 1.1083 - accuracy: 0.66
39 - val_loss: 1.1012 - val_accuracy: 0.6644
Epoch 30/30
108/108 [==============================] - 31s 289ms/step - loss: 1.1015 - accuracy: 0.66
53 - val_loss: 1.0956 - val_accuracy: 0.6667
```

Out[ ]:

```
<keras.callbacks.History at 0x7f4e0dcc4dd0>
```

In [ ]:

```
# Print prediction(s)
print("Prediction:")
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))

print("\nCorrect Translation:")
print(french_sentences[:1])

print("\nOriginal text:")
print(english_sentences[:1])
```

```
Prediction:
new jersey est parfois calme en l' et il est est en en <PAD> <PAD> <PAD> <PAD> <PAD> <PAD
> <PAD> <PAD>

Correct Translation:
["new jersey est parfois calme pendant l' automne , et il est neigeux en avril ."]

Original text:
['new jersey is sometimes quiet during autumn , and it is snowy in april .']
```

In [ ]:

```
#changing the number of epochs from 30 to 40
# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024, epochs=40, valida
tion_split=0.2)
```

```
Epoch 1/40
108/108 [==============================] - 34s 295ms/step - loss: 4.5155 - accuracy: 0.35
72 - val_loss: 2.5159 - val_accuracy: 0.4691
Epoch 2/40
108/108 [==============================] - 31s 288ms/step - loss: 2.4554 - accuracy: 0.47
05 - val_loss: 2.2671 - val_accuracy: 0.5021
Epoch 3/40
108/108 [==============================] - 31s 285ms/step - loss: 2.2051 - accuracy: 0.51
39 - val_loss: 2.0105 - val_accuracy: 0.5528
Epoch 4/40
108/108 [==============================] - 31s 288ms/step - loss: 1.9596 - accuracy: 0.55
```

67 - val_loss: 1.8215 - val_accuracy: 0.5774
Epoch 5/40
108/108 [==============================] - 31s 288ms/step - loss: 1.7846 - accuracy: 0.57
74 - val_loss: 1.6899 - val_accuracy: 0.5834
Epoch 6/40
108/108 [==============================] - 31s 287ms/step - loss: 1.6682 - accuracy: 0.58
43 - val_loss: 1.5977 - val_accuracy: 0.5914
Epoch 7/40
108/108 [==============================] - 31s 286ms/step - loss: 1.5813 - accuracy: 0.59
13 - val_loss: 1.5281 - val_accuracy: 0.5981
Epoch 8/40
108/108 [==============================] - 31s 285ms/step - loss: 1.5139 - accuracy: 0.60
45 - val_loss: 1.4737 - val_accuracy: 0.6152
Epoch 9/40
108/108 [==============================] - 30s 282ms/step - loss: 1.4633 - accuracy: 0.61
57 - val_loss: 1.4286 - val_accuracy: 0.6191
Epoch 10/40
108/108 [==============================] - 31s 283ms/step - loss: 1.4204 - accuracy: 0.61
93 - val_loss: 1.3890 - val_accuracy: 0.6262
Epoch 11/40
108/108 [==============================] - 30s 282ms/step - loss: 1.3796 - accuracy: 0.62
69 - val_loss: 1.3533 - val_accuracy: 0.6327
Epoch 12/40
108/108 [==============================] - 30s 282ms/step - loss: 1.3454 - accuracy: 0.63
31 - val_loss: 1.3217 - val_accuracy: 0.6368
Epoch 13/40
108/108 [==============================] - 31s 283ms/step - loss: 1.3168 - accuracy: 0.63
84 - val_loss: 1.2935 - val_accuracy: 0.6438
Epoch 14/40
108/108 [==============================] - 31s 284ms/step - loss: 1.2876 - accuracy: 0.64
53 - val_loss: 1.2689 - val_accuracy: 0.6462
Epoch 15/40
108/108 [==============================] - 31s 287ms/step - loss: 1.2634 - accuracy: 0.64
78 - val_loss: 1.2464 - val_accuracy: 0.6488
Epoch 16/40
108/108 [==============================] - 31s 284ms/step - loss: 1.2421 - accuracy: 0.65
07 - val_loss: 1.2292 - val_accuracy: 0.6512
Epoch 17/40
108/108 [==============================] - 31s 285ms/step - loss: 1.2249 - accuracy: 0.65
25 - val_loss: 1.2096 - val_accuracy: 0.6545
Epoch 18/40
108/108 [==============================] - 31s 286ms/step - loss: 1.2094 - accuracy: 0.65
36 - val_loss: 1.1953 - val_accuracy: 0.6550
Epoch 19/40
108/108 [==============================] - 31s 287ms/step - loss: 1.1909 - accuracy: 0.65
68 - val_loss: 1.1803 - val_accuracy: 0.6562
Epoch 20/40
108/108 [==============================] - 31s 289ms/step - loss: 1.1763 - accuracy: 0.65
83 - val_loss: 1.1681 - val_accuracy: 0.6582
Epoch 21/40
108/108 [==============================] - 31s 288ms/step - loss: 1.1652 - accuracy: 0.66
00 - val_loss: 1.1560 - val_accuracy: 0.6584
Epoch 22/40
108/108 [==============================] - 31s 285ms/step - loss: 1.1525 - accuracy: 0.66
11 - val_loss: 1.1447 - val_accuracy: 0.6609
Epoch 23/40
108/108 [==============================] - 31s 286ms/step - loss: 1.1448 - accuracy: 0.66
12 - val_loss: 1.1372 - val_accuracy: 0.6612
Epoch 24/40
108/108 [==============================] - 31s 284ms/step - loss: 1.1348 - accuracy: 0.66
21 - val_loss: 1.1254 - val_accuracy: 0.6667
Epoch 25/40
108/108 [==============================] - 31s 284ms/step - loss: 1.1240 - accuracy: 0.66
51 - val_loss: 1.1173 - val_accuracy: 0.6650
Epoch 26/40
108/108 [==============================] - 31s 284ms/step - loss: 1.1157 - accuracy: 0.66
57 - val_loss: 1.1076 - val_accuracy: 0.6659
Epoch 27/40
108/108 [==============================] - 31s 283ms/step - loss: 1.1048 - accuracy: 0.66
81 - val_loss: 1.0999 - val_accuracy: 0.6661
Epoch 28/40
108/108 [==============================] - 30s 281ms/step - loss: 1.0983 - accuracy: 0.66

```
79 - val_loss: 1.0919 - val_accuracy: 0.6664
Epoch 29/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0929 - accuracy: 0.66
79 - val_loss: 1.0853 - val_accuracy: 0.6690
Epoch 30/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0856 - accuracy: 0.66
85 - val_loss: 1.0799 - val_accuracy: 0.6682
Epoch 31/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0760 - accuracy: 0.67
05 - val_loss: 1.0714 - val_accuracy: 0.6710
Epoch 32/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0715 - accuracy: 0.67
12 - val_loss: 1.0650 - val_accuracy: 0.6678
Epoch 33/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0633 - accuracy: 0.67
22 - val_loss: 1.0590 - val_accuracy: 0.6713
Epoch 34/40
108/108 [==============================] - 30s 276ms/step - loss: 1.0568 - accuracy: 0.67
30 - val_loss: 1.0538 - val_accuracy: 0.6759
Epoch 35/40
108/108 [==============================] - 30s 281ms/step - loss: 1.0506 - accuracy: 0.67
60 - val_loss: 1.0476 - val_accuracy: 0.6730
Epoch 36/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0415 - accuracy: 0.67
69 - val_loss: 1.0421 - val_accuracy: 0.6800
Epoch 37/40
108/108 [==============================] - 30s 281ms/step - loss: 1.0403 - accuracy: 0.67
76 - val_loss: 1.0365 - val_accuracy: 0.6776
Epoch 38/40
108/108 [==============================] - 30s 279ms/step - loss: 1.0343 - accuracy: 0.67
73 - val_loss: 1.0300 - val_accuracy: 0.6729
Epoch 39/40
108/108 [==============================] - 31s 283ms/step - loss: 1.0289 - accuracy: 0.67
72 - val_loss: 1.0244 - val_accuracy: 0.6783
Epoch 40/40
108/108 [==============================] - 31s 284ms/step - loss: 1.0238 - accuracy: 0.67
98 - val_loss: 1.0217 - val_accuracy: 0.6854
```

Out[ ]:

```
<keras.callbacks.History at 0x7f956e3b2b10>
```

In [ ]:

```
# Print prediction(s)
print("Prediction:")
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))

print("\nCorrect Translation:")
print(french_sentences[:1])

print("\nOriginal text:")
print(english_sentences[:1])
```

```
Prediction:
new jersey est parfois calme en l' et il est est en agréable <PAD> <PAD> <PAD> <PAD> <PAD
> <PAD> <PAD> <PAD>

Correct Translation:
["new jersey est parfois calme pendant l' automne , et il est neigeux en avril ."]

Original text:
['new jersey is sometimes quiet during autumn , and it is snowy in april .']
```

In [ ]:

```
#changing the number of epochs from 40 to 50
# Train the neural network
simple_rnn_model = simple_model(
    tmp_x.shape,
    max_french_sequence_length,
    english_vocab_size,
```

```
    french_vocab_size)
simple_rnn_model.fit(tmp_x, preproc_french_sentences, batch_size=1024, epochs=50, valida
tion_split=0.2)
```

```
Epoch 1/50
108/108 [==============================] - 35s 304ms/step - loss: 4.3963 - accuracy: 0.37
57 - val_loss: 2.5702 - val_accuracy: 0.4543
Epoch 2/50
108/108 [==============================] - 32s 296ms/step - loss: 2.5150 - accuracy: 0.45
97 - val_loss: 2.3439 - val_accuracy: 0.4718
Epoch 3/50
108/108 [==============================] - 31s 289ms/step - loss: 2.2826 - accuracy: 0.47
96 - val_loss: 2.0830 - val_accuracy: 0.5262
Epoch 4/50
108/108 [==============================] - 31s 290ms/step - loss: 2.0156 - accuracy: 0.53
49 - val_loss: 1.8479 - val_accuracy: 0.5686
Epoch 5/50
108/108 [==============================] - 31s 291ms/step - loss: 1.8110 - accuracy: 0.56
96 - val_loss: 1.7217 - val_accuracy: 0.5742
Epoch 6/50
108/108 [==============================] - 32s 295ms/step - loss: 1.6997 - accuracy: 0.57
62 - val_loss: 1.6398 - val_accuracy: 0.5851
Epoch 7/50
108/108 [==============================] - 31s 292ms/step - loss: 1.6224 - accuracy: 0.58
71 - val_loss: 1.5764 - val_accuracy: 0.5950
Epoch 8/50
108/108 [==============================] - 32s 296ms/step - loss: 1.5626 - accuracy: 0.59
56 - val_loss: 1.5241 - val_accuracy: 0.6001
Epoch 9/50
108/108 [==============================] - 32s 294ms/step - loss: 1.5107 - accuracy: 0.60
27 - val_loss: 1.4784 - val_accuracy: 0.6092
Epoch 10/50
108/108 [==============================] - 32s 294ms/step - loss: 1.4683 - accuracy: 0.61
29 - val_loss: 1.4384 - val_accuracy: 0.6149
Epoch 11/50
108/108 [==============================] - 31s 290ms/step - loss: 1.4278 - accuracy: 0.62
26 - val_loss: 1.4042 - val_accuracy: 0.6285
Epoch 12/50
108/108 [==============================] - 31s 287ms/step - loss: 1.3970 - accuracy: 0.62
99 - val_loss: 1.3745 - val_accuracy: 0.6315
Epoch 13/50
108/108 [==============================] - 31s 288ms/step - loss: 1.3682 - accuracy: 0.63
26 - val_loss: 1.3474 - val_accuracy: 0.6354
Epoch 14/50
108/108 [==============================] - 31s 287ms/step - loss: 1.3387 - accuracy: 0.63
72 - val_loss: 1.3209 - val_accuracy: 0.6401
Epoch 15/50
108/108 [==============================] - 31s 287ms/step - loss: 1.3130 - accuracy: 0.64
10 - val_loss: 1.2964 - val_accuracy: 0.6436
Epoch 16/50
108/108 [==============================] - 31s 291ms/step - loss: 1.2921 - accuracy: 0.64
34 - val_loss: 1.2749 - val_accuracy: 0.6446
Epoch 17/50
108/108 [==============================] - 31s 292ms/step - loss: 1.2681 - accuracy: 0.64
66 - val_loss: 1.2516 - val_accuracy: 0.6479
Epoch 18/50
108/108 [==============================] - 32s 297ms/step - loss: 1.2492 - accuracy: 0.64
86 - val_loss: 1.2316 - val_accuracy: 0.6509
Epoch 19/50
108/108 [==============================] - 33s 301ms/step - loss: 1.2258 - accuracy: 0.65
28 - val_loss: 1.2143 - val_accuracy: 0.6522
Epoch 20/50
108/108 [==============================] - 32s 294ms/step - loss: 1.2109 - accuracy: 0.65
38 - val_loss: 1.1986 - val_accuracy: 0.6582
Epoch 21/50
108/108 [==============================] - 32s 292ms/step - loss: 1.1964 - accuracy: 0.65
66 - val_loss: 1.1845 - val_accuracy: 0.6580
Epoch 22/50
108/108 [==============================] - 32s 295ms/step - loss: 1.1801 - accuracy: 0.65
70 - val_loss: 1.1724 - val_accuracy: 0.6579
Epoch 23/50
108/108 [==============================] - 32s 293ms/step - loss: 1.1701 - accuracy: 0.65
```

```
83 - val_loss: 1.1603 - val_accuracy: 0.6589
Epoch 24/50
108/108 [==============================] - 32s 294ms/step - loss: 1.1594 - accuracy: 0.65
86 - val_loss: 1.1485 - val_accuracy: 0.6606
Epoch 25/50
108/108 [==============================] - 32s 293ms/step - loss: 1.1469 - accuracy: 0.66
06 - val_loss: 1.1399 - val_accuracy: 0.6603
Epoch 26/50
108/108 [==============================] - 31s 290ms/step - loss: 1.1391 - accuracy: 0.66
05 - val_loss: 1.1315 - val_accuracy: 0.6615
Epoch 27/50
108/108 [==============================] - 32s 294ms/step - loss: 1.1310 - accuracy: 0.66
31 - val_loss: 1.1225 - val_accuracy: 0.6626
Epoch 28/50
108/108 [==============================] - 31s 290ms/step - loss: 1.1200 - accuracy: 0.66
46 - val_loss: 1.1151 - val_accuracy: 0.6648
Epoch 29/50
108/108 [==============================] - 32s 292ms/step - loss: 1.1159 - accuracy: 0.66
49 - val_loss: 1.1073 - val_accuracy: 0.6680
Epoch 30/50
108/108 [==============================] - 31s 288ms/step - loss: 1.1062 - accuracy: 0.66
72 - val_loss: 1.1015 - val_accuracy: 0.6672
Epoch 31/50
108/108 [==============================] - 31s 289ms/step - loss: 1.1021 - accuracy: 0.66
65 - val_loss: 1.0944 - val_accuracy: 0.6708
Epoch 32/50
108/108 [==============================] - 31s 289ms/step - loss: 1.0938 - accuracy: 0.66
89 - val_loss: 1.0888 - val_accuracy: 0.6709
Epoch 33/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0893 - accuracy: 0.66
87 - val_loss: 1.0836 - val_accuracy: 0.6721
Epoch 34/50
108/108 [==============================] - 31s 289ms/step - loss: 1.0848 - accuracy: 0.66
99 - val_loss: 1.0772 - val_accuracy: 0.6679
Epoch 35/50
108/108 [==============================] - 31s 289ms/step - loss: 1.0748 - accuracy: 0.67
17 - val_loss: 1.0713 - val_accuracy: 0.6729
Epoch 36/50
108/108 [==============================] - 31s 287ms/step - loss: 1.0703 - accuracy: 0.67
35 - val_loss: 1.0680 - val_accuracy: 0.6747
Epoch 37/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0647 - accuracy: 0.67
55 - val_loss: 1.0604 - val_accuracy: 0.6718
Epoch 38/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0582 - accuracy: 0.67
68 - val_loss: 1.0551 - val_accuracy: 0.6785
Epoch 39/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0509 - accuracy: 0.68
02 - val_loss: 1.0486 - val_accuracy: 0.6787
Epoch 40/50
108/108 [==============================] - 31s 287ms/step - loss: 1.0457 - accuracy: 0.68
19 - val_loss: 1.0413 - val_accuracy: 0.6816
Epoch 41/50
108/108 [==============================] - 31s 286ms/step - loss: 1.0397 - accuracy: 0.68
46 - val_loss: 1.0354 - val_accuracy: 0.6864
Epoch 42/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0333 - accuracy: 0.68
63 - val_loss: 1.0301 - val_accuracy: 0.6890
Epoch 43/50
108/108 [==============================] - 31s 288ms/step - loss: 1.0309 - accuracy: 0.68
82 - val_loss: 1.0247 - val_accuracy: 0.6871
Epoch 44/50
108/108 [==============================] - 31s 289ms/step - loss: 1.0223 - accuracy: 0.68
95 - val_loss: 1.0204 - val_accuracy: 0.6881
Epoch 45/50
108/108 [==============================] - 31s 286ms/step - loss: 1.0184 - accuracy: 0.69
06 - val_loss: 1.0159 - val_accuracy: 0.6904
Epoch 46/50
108/108 [==============================] - 31s 289ms/step - loss: 1.0161 - accuracy: 0.68
96 - val_loss: 1.0106 - val_accuracy: 0.6906
Epoch 47/50
108/108 [==============================] - 31s 287ms/step - loss: 1.0085 - accuracy: 0.69
```

```
25 - val_loss: 1.0065 - val_accuracy: 0.6946
Epoch 48/50
108/108 [==============================] - 31s 287ms/step - loss: 1.0045 - accuracy: 0.69
41 - val_loss: 1.0024 - val_accuracy: 0.6993
Epoch 49/50
108/108 [==============================] - 31s 289ms/step - loss: 0.9993 - accuracy: 0.69
81 - val_loss: 0.9974 - val_accuracy: 0.6999
Epoch 50/50
108/108 [==============================] - 31s 287ms/step - loss: 0.9948 - accuracy: 0.69
89 - val_loss: 0.9957 - val_accuracy: 0.6940
```

Out[ ]:

```
<keras.callbacks.History at 0x7f956e48a690>
```

In [ ]:

```
# Print prediction(s)
print("Prediction:")
print(logits_to_text(simple_rnn_model.predict(tmp_x[:1])[0], french_tokenizer))

print("\nCorrect Translation:")
print(french_sentences[:1])

print("\nOriginal text:")
print(english_sentences[:1])
```

```
Prediction:
new jersey est parfois calme en mois et il et il en agréable <PAD> <PAD> <PAD> <PAD> <PAD
> <PAD> <PAD> <PAD>

Correct Translation:
["new jersey est parfois calme pendant l' automne , et il est neigeux en avril ."]

Original text:
['new jersey is sometimes quiet during autumn , and it is snowy in april .']
```