Text Translation in Python

What are we going to do in this video?

In this video, we will build a text translation system in python which will translate a text given in english to french.

So, we will be using a data set of English to French which contains list of english sentences and their corresponding list of french translations.

The prerequisite of this video is that you should be curious about how neural networks work for text translation but at the same time the main focus would be on coding the text translation algo rather than going to much deep into the theory behind how these algos work, so we will focus on the interesting part more.

How this video will be laid out?

To build the whole translator in one go can get quite tricky, but we are going to be following a step by step approach so that we don't get lost in any step. Each step will have a corresponding pseudo code associated with it.

1. Importing the necessary modules.
2. Acquiring the dataset.
3. Building an Encoder Model
4. Building a Decoder Model
5. Wrapping Encoder and Decoder in a Model with Dense and Time Distributed Layer
6. Compiling the model
7. Building Tokenizer
8. Building Preprocessing Functions for english and french words
9. Creating the data set from the preprocessing Functions.
10. Training the Neural Network and Making Prediction.

Pseudocode for building the text Translation system

Importing the Libraries

```
import requests

import tensorflow.keras as keras
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import Dense, TimeDistributed
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical

from tensorflow.keras.preprocessing.text import Tokenizer

import numpy as np
```

Acquiring the dataset

```
# Define two variables for URLs
french_url
english_url

# Make a request to those above URLs and get the text
french_sentences
english_sentences

# Split the text with new line
french_sentences = french_sentences.split("\n")
english_sentences = english_sentences.split("\n")
```

Encoder

```
# Define some important variables like word length, vocan length for
# Both english and french

en_len = 20
en_vocab = 100
fr_len = 25
fr_vocab = 125
hsize = 48

# Define the input to the encoder which is going to be english sentences
encoder_input = keras.layers.Input(shape = (en_len, en_vocab))

# Define Encoder output and state using GRU layer. Encoder state will be used
# as a context vector for the decoder

encoder_output, encoder_state = Make_a_GRU_layer with hsize and return_state = True
```

Decoder

```
# Repeat the context vector produced by the encoder
decoder_input = RepeatVector(length of french sentence)(encoder_state)

# Get the decoder GRU output
decoder_gru_output = GRU(hsize, return_state = True)(input, set initial_state )
```

Dense and Time Distributed Layers

```
# Now, define a dense layer with fr_vocab number of neurons and softmax as
# the activation function and call it decoder_dense

decoder_dense = Dense()

# Wrap the dense layer with Time Distributed layer to produce sequences

decoder_dense_time = TimeDistributed(decoder_dense)

# Now, define the input of the decoder_dense_time layer as decoder_gru_output
# Because whatever decoder_gru_output returns will be fed to the time
# distributed layer.

decoder_prediction = decoder_dense_time(decoder_gru_output)
```

Model and compilation

```
# Now, build a model using Model class by specifying
# the input and outputs
model = Model(inputs = encoder_input, outputs = decoder_prediction)

# Finally, compile the model by setting
# optimizer = "adam", loss as "categorical_crossentropy" and metrics as "acc"
model.compile()

# To confirm, check the summary of the model
```

Tokenization models

```
# Now, initiaze a Tokenizer for both english and french and set out of
# vocabulary words as "UNK"
english_token = Tokenizer(num_words = en_vocab)
french_token = Tokenizer(num_words = fr_vocab)

# Fit the tokenizer on their respective texts
english_token.fit_on_texts()
french_token.fit_on_texts()
```

Preprocessing the data (Helper functions)

```
# To build the helper functions
def preprocess_input(sentences):
  # First get encoded text from
  # english_token.texts_to_sequences(sentences)
  encoded_text
  # Then call the pad_sequences() on the encoded_text with
  # padding and trucncating = "post" and max_len = sentence length
  preprocess_text
  now, since it is input, reverse the text along the second dimension
  # and finally convert the text in one hot encoding using to_categorical
  final_preprocessed_text = to_categorical(preprocess_text)
  # with num_classes=vocab_size
  return final_preprocessed_text

# Do the same for french but without reversing the texts
```

Finally, using the functions and training the

```
# Get the preprocessed english and french sentences
english_X
french_y

# Specify the appropriate number of epochs and batch_size
# and fit the model

model.fit(english_X, french_y, epochs = 100, batch_size = 1000)
```

Now, let us make some predictions

Thank you