```
import pandas as pd
import re
import tensorflow as tf
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
import nltk.translate.bleu_score as bleu
import random
import string
from sklearn.model_selection import train_test_split
import os
import time
```

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

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

```
eng_hin=pd.read_csv('/content/drive/MyDrive/PRL/task/data/synthetic-dataset/train.csv')
eng_hin.head()
```

	English	Hindi	Hinglish	Average rating	Disagreement
0	Program module is a file that contains instruc...	माड्यूल, एक संचिका होती है, जिसमें या तो स्रोत...	module , ek program hoti hai , jismen ya to so...	7	6
1	And to Thamud We sent their brother Sali 'h. H...	और (हमने) क़ौमे समूद के पास उनके भाई सालेह को ...	aur hamne aume samood ke pas unke bhaee saleh ...	6	4
2	and, when reminded, do not remember\n	और जब उन्हें याद दिलाया जाता है, तो वे याद नही...	aur jab unhen yad dilaya jata hai , to ve yad ...	10	0
		तम्हें २०११ का टेड			

```
# create a new dataframe of english and hinglish column
df = pd.DataFrame()
df["english"] = eng_hin["English"]
df["hindi"] = eng_hin["Hinglish"]
df.head()
```

	english	hindi
0	Program module is a file that contains instruc...	module , ek program hoti hai , jismen ya to so...
1	And to Thamud We sent their brother Sali 'h. H...	aur hamne aume samood ke pas unke bhaee saleh ...
2	and, when reminded, do not remember\n	aur jab unhen yad dilaya jata hai , to ve yad ...
3	you won the TED Prize 2011.\n	tumhen २०११ ka ted prize mil gaya hai\n
4	He gone to Kerodemal College of Delhi Universi...	unhonne bad science karne ke lie ye delhi univ...

```
eng_hin.dropna(inplace=True)
eng_hin.shape
```

```
(2766, 5)
```

```
exclude = set(string.punctuation) # Set of all special characters
remove_digits = str.maketrans('', '', string.digits) # Set of all digits
```

```
def preprocess(text):
    '''Function to preprocess English sentence'''
    text = text.lower() # lower casing
    text = re.sub("'", '', text) # remove the quotation marks if any
    text = ''.join(ch for ch in text if ch not in exclude)
    text = text.translate(remove_digits) # remove the digits
    text = text.strip()
    text = re.sub(" +", " ", text) # remove extra spaces
    text = '<start> ' + text + ' <end>'
    return text
```

```
def preprocess_hin(text):
    '''Function to preprocess Marathi sentence'''
    text = re.sub("'", '', text) # remove the quotation marks if any
    text = ''.join(ch for ch in text if ch not in exclude)
    text = re.sub("[२३०८१५७९४६]", "", text) # remove the digits
    text = text.strip()
    text = re.sub(" +", " ", text) # remove extra spaces
    text = '<start> ' + text + ' <end>'
    return text
```

```
eng_hin['english'] = df['english'].apply(preprocess)
eng_hin['hindi'] = df['hindi'].apply(preprocess_hin)

eng_hin.rename(columns={"english": "english", "hindi": "hindi"},inplace=True)

eng_hin.head()
```

	English	Hindi	Hinglish	Average rating	Disagreement	english	hindi
0	Program module is a file that contains instruc...	माड्यूल, एक संचिका होती है, जिसमें या तो स्रोत...	module , ek program hoti hai , jismen ya to so...	7	6	<start> program module is a file that contains...	<start> module ek program hoti hai jismen ya t...
1	And to Thamud We sent their brother Sali 'h. H...	और (हमने) क़ौमे समूद के पास उनके भाई सालेह को ...	aur hamne aume samood ke pas unke bhaee saleh ...	6	4	<start> and to thamud we sent their brother sa...	<start> aur hamne aume samood ke pas unke bhae...
2	and, when reminded, do not remember\n	और जब उन्हें याद दिलाया जाता है, तो वे याद नही...	aur jab unhen yad dilaya jata hai , to ve yad ...	10	0	<start> and when reminded do not remember <end>	<start> aur jab unhen yad dilaya jata hai to v...
3	you won the TED Prize 2011.\n	तुम्हें २०११ का टेड प्राइज़ मिल गया है.\n	tumhen २०११ ka ted prize mil gaya hai\n	9	1	<start> you won the ted prize <end>	<start> tumhen ka ted prize mil gaya hai <end>
	He gone to	उन्होंने बाद	unhonne bad			<start> he gone to	<start> unhonne

```
def tokenize(lang):

  lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
  lang_tokenizer.fit_on_texts(lang)

  tensor = lang_tokenizer.texts_to_sequences(lang)

  tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor,padding='post',maxlen=20,dtype='int32')

  return tensor, lang_tokenizer
```

```
def load_dataset():

  input_tensor, inp_lang_tokenizer = tokenize(eng_hin['english'].values)
  target_tensor, targ_lang_tokenizer = tokenize(eng_hin['hindi'].values)

  return input_tensor, target_tensor, inp_lang_tokenizer, targ_lang_tokenizer
```

```
input_tensor, target_tensor, inp_lang, targ_lang = load_dataset()
```

```
max_length_targ, max_length_inp = target_tensor.shape[1], input_tensor.shape[1]
```

```
input_tensor_train, input_tensor_val, target_tensor_train, target_tensor_val = train_test_split(input_tensor, target_te

print(len(input_tensor_train), len(target_tensor_train), len(input_tensor_val), len(target_tensor_val))
```

```
2212 2212 554 554
```

```python
BUFFER_SIZE = len(input_tensor_train)
BATCH_SIZE = 32
N_BATCH = BUFFER_SIZE//BATCH_SIZE
embedding_dim = 256
units = 1024
steps_per_epoch = len(input_tensor_train)//BATCH_SIZE

vocab_inp_size =len(inp_lang.word_index.keys())
vocab_tar_size =len(targ_lang.word_index.keys())

dataset = tf.data.Dataset.from_tensor_slices((input_tensor_train, target_tensor_train)).shuffle(BUFFER_SIZE)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
```

```python
embeddings_index = dict()
f = open('/content/drive/MyDrive/PRL/task/data/synthetic-dataset/glove.6B.300d.txt')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

embedding_matrix = np.zeros((vocab_inp_size+1, 300))
for word, i in inp_lang.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```python
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(input_dim=vocab_size, output_dim=embedding_dim, name="embedding_laye
        self.gru = tf.keras.layers.GRU(units, return_sequences=True, return_state=True, recurrent_activation='sigmoid',

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        return output, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```

```python
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(units, return_sequences=True, return_state=True, recurrent_activation='sigmoid',
        self.fc = tf.keras.layers.Dense(vocab_size)

                # used for attention
        self.W1 = tf.keras.layers.Dense(self.dec_units)
        self.W2 = tf.keras.layers.Dense(self.dec_units)
        self.V = tf.keras.layers.Dense(1)

    def call(self, x, hidden, enc_output):

        hidden_with_time_axis = tf.expand_dims(hidden, 1)

        score = self.V(tf.nn.tanh(self.W1(enc_output) + self.W2(hidden_with_time_axis)))

        attention_weights = tf.nn.softmax(score, axis=1)

        context_vector = attention_weights * enc_output
        context_vector = tf.reduce_sum(context_vector, axis=1)

        x = self.embedding(x)

        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        output, state = self.gru(x)

        output = tf.reshape(output, (-1, output.shape[2]))

        x = self.fc(output)

        return x, state, attention_weights

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.dec_units))
```

```
tf.keras.backend.clear_session()

encoder = Encoder(vocab_inp_size+1, 300, units, BATCH_SIZE)
decoder = Decoder(vocab_tar_size+1, embedding_dim, units, BATCH_SIZE)
```

```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
                                                            reduction='none')


def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)

  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask

  return tf.reduce_mean(loss_)
```

```
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(optimizer=optimizer,
                                 encoder=encoder,
                                 decoder=decoder)
```

```
@tf.function
def train_step(inp, targ, enc_hidden):
  loss = 0

  with tf.GradientTape() as tape:
    enc_output, enc_hidden = encoder(inp, enc_hidden)
    encoder.get_layer('embedding_layer_encoder').set_weights([embedding_matrix])
    dec_hidden = enc_hidden

    dec_input = tf.expand_dims([targ_lang.word_index['<start>']] * BATCH_SIZE, 1)

    for t in range(1, targ.shape[1]):
      predictions, dec_hidden, _ = decoder(dec_input, dec_hidden, enc_output)

      loss += loss_function(targ[:, t], predictions)

      dec_input = tf.expand_dims(targ[:, t], 1)

  batch_loss = (loss / int(targ.shape[1]))

  variables = encoder.trainable_variables + decoder.trainable_variables

  gradients = tape.gradient(loss, variables)

  optimizer.apply_gradients(zip(gradients, variables))

  return batch_loss
```

```
EPOCHS = 100

for epoch in range(EPOCHS):
  start = time.time()

  enc_hidden = encoder.initialize_hidden_state()
  total_loss = 0

  for (batch, (inp, targ)) in enumerate(dataset.take(steps_per_epoch)):
    batch_loss = train_step(inp, targ, enc_hidden)
    total_loss += batch_loss

    if batch % 100 == 0:
      print(f'Epoch {epoch+1} Batch {batch} Loss {batch_loss.numpy():.4f}')
  if (epoch + 1) % 2 == 0:
    checkpoint.save(file_prefix=checkpoint_prefix)

  print(f'Epoch {epoch+1} Loss {total_loss/steps_per_epoch:.4f}')
  print(f'Time taken for 1 epoch {time.time()-start:.2f} sec\n')
```

```
Time taken for 1 epoch 8.22 sec

Epoch 87 Batch 0 Loss 0.0370
Epoch 87 Loss 0.0708
Time taken for 1 epoch 7.46 sec

Epoch 88 Batch 0 Loss 0.0429
Epoch 88 Loss 0.0708
Time taken for 1 epoch 8.18 sec

Epoch 89 Batch 0 Loss 0.0513
Epoch 89 Loss 0.0706
Time taken for 1 epoch 7.34 sec

Epoch 90 Batch 0 Loss 0.0318
Epoch 90 Loss 0.0705
Time taken for 1 epoch 8.17 sec

Epoch 91 Batch 0 Loss 0.0576
Epoch 91 Loss 0.0697
Time taken for 1 epoch 7.75 sec

Epoch 92 Batch 0 Loss 0.0290
Epoch 92 Loss 0.0709
Time taken for 1 epoch 8.14 sec

Epoch 93 Batch 0 Loss 0.0380
Epoch 93 Loss 0.0697
Time taken for 1 epoch 7.37 sec

Epoch 94 Batch 0 Loss 0.0457
Epoch 94 Loss 0.0694
Time taken for 1 epoch 8.17 sec

Epoch 95 Batch 0 Loss 0.0720
Epoch 95 Loss 0.0711
Time taken for 1 epoch 7.35 sec

Epoch 96 Batch 0 Loss 0.0363
Epoch 96 Loss 0.0763
Time taken for 1 epoch 8.23 sec

Epoch 97 Batch 0 Loss 0.0619
Epoch 97 Loss 0.0799
Time taken for 1 epoch 7.34 sec

Epoch 98 Batch 0 Loss 0.0530
Epoch 98 Loss 0.0859
Time taken for 1 epoch 8.18 sec

Epoch 99 Batch 0 Loss 0.0924
Epoch 99 Loss 0.1036
Time taken for 1 epoch 7.31 sec

Epoch 100 Batch 0 Loss 0.0458
Epoch 100 Loss 0.1193
Time taken for 1 epoch 8.32 sec
```

```
def evaluate(sentence):
  attention_plot = np.zeros((max_length_targ, max_length_inp))

  sentence = preprocess(sentence)

  inputs = [inp_lang.word_index[i] for i in sentence.split(' ')]
  inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],maxlen=20, padding='post')
  inputs = tf.convert_to_tensor(inputs)

  result = ''

  hidden = [tf.zeros((1, units))]
  enc_out, enc_hidden = encoder(inputs, hidden)

  dec_hidden = enc_hidden
  dec_input = tf.expand_dims([targ_lang.word_index['<start>']], 0)

  for t in range(max_length_targ):
    predictions, dec_hidden, attention_weights = decoder(dec_input,
                                                         dec_hidden,
                                                         enc_out)
    # storing the attention weights to plot later on
    attention_weights = tf.reshape(attention_weights, (-1, ))
    attention_plot[t] = attention_weights.numpy()
    predicted_id = tf.argmax(predictions[0]).numpy()

    result += targ_lang.index_word[predicted_id] + ' '

    if targ_lang.index_word[predicted_id] == '<end>':
      return result,attention_plot

    # the predicted ID is fed back into the model
    dec_input = tf.expand_dims([predicted_id], 0)

  return result,attention_plot
```

```
input_sentence= 'please ensure that you use the appropriate form '
print('Input sentence in english : ',input_sentence)
predicted_output,attention_plot=evaluate(input_sentence)
print('Predicted sentence in hindi : ',predicted_output)
```

```
Input sentence in english :  please ensure that you use the appropriate form
Predicted sentence in hindi :  check len ki spelling karna hai <end>
```

```
input_sentence='and do something with it to change the world '
print('Input sentence in english : ',input_sentence)
predicted_output,attention_plot=evaluate(input_sentence)
print('Predicted sentence in hindi : ',predicted_output)
```

```
Input sentence in english :  and do something with it to change the world
Predicted sentence in hindi :  aur ye kuchh ve in logon ke samne ek sath same <end>
```

OpenInAPP sentences output

```
input_sentence='So even if its a big video I will clearly mention all the products '
print('Input sentence in english : ',input_sentence)
predicted_output,attention_plot=evaluate(input_sentence)
print('Predicted sentence in hindi : ',predicted_output)
```

```
Input sentence in english :  So even if its a big video I will clearly mention all the products
Predicted sentence in hindi :  ata ham yah bhi likha to parishram men aap basis par aap basis par aap basis par aap ba
```

```
input_sentence='I was waiting for my bag '
print('Input sentence in english : ',input_sentence)
predicted_output,attention_plot=evaluate(input_sentence)
print('Predicted sentence in hindi : ',predicted_output)
```

```
Input sentence in english :  I was waiting for my bag
Predicted sentence in hindi :  i main ek ball pap kiya gaya tha <end>
```

```
  File "<ipython-input-34-e8c0dc1d82b3>", line 1
    Definitely share your feedback in the comment section
               ^
```

```
input_sentence='definitely share your feedback in the comment section '
print('Input sentence in english : ',input_sentence)
predicted_output,attention_plot=evaluate(input_sentence)
print('Predicted sentence in hindi : ',predicted_output)
```

```
Input sentence in english :  definitely share your feedback in the comment section
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-40-854b97a97912> in <cell line: 3>()
      1 input_sentence='definitely share your feedback in the comment section '
      2 print('Input sentence in english : ',input_sentence)
----> 3 predicted_output,attention_plot=evaluate(input_sentence)
      4 print('Predicted sentence in hindi : ',predicted_output)

                                 1 frames
<ipython-input-24-4ac5775c6a63> in evaluate(sentence)
      4     sentence = preprocess(sentence)
      5 
----> 6     inputs = [inp_lang.word_index[i] for i in sentence.split(' ')]
      7     inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],maxlen=20, padding='post')
      8     inputs = tf.convert_to_tensor(inputs)

<ipython-input-24-4ac5775c6a63> in <listcomp>(.0)
      4     sentence = preprocess(sentence)
      5 
----> 6     inputs = [inp_lang.word_index[i] for i in sentence.split(' ')]
      7     inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],maxlen=20, padding='post')
      8     inputs = tf.convert_to_tensor(inputs)

KeyError: 'definitely'
```

SEARCH STACK OVERFLOW

Colab paid products - Cancel contracts here

0s completed at 9:19 PM