```
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
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
4	He gone to Kerodemal College of Delhi Universi...	उन्होंने बाद अध्ययन करने के लिए ये दिल्ली विश्...	unhonne bad science karne ke lie ye delhi univ...	7	0

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

	english	hindi
0	Program module is a file that contains instruc...	माड्यूल, एक संचिका होती है, जिसमें या तो स्रोत...
1	And to Thamud We sent their brother Sali 'h. H...	और (हमने) क़ौमे समूद के पास उनके भाई सालेह को ...
2	and, when reminded, do not remember\n	और जब उन्हें याद दिलाया जाता है, तो वे याद नही...
3	you won the TED Prize 2011.\n	तुम्हें २०११ का टेड प्राइज़ मिल गया है.\n
4	He gone to Kerodemal College of Delhi Universi...	उन्होंने बाद अध्ययन करने के लिए ये दिल्ली विश्...

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

```
(2766, 7)
```

```
import pickle

with open("/content/drive/MyDrive/PRL/task/data/human-generated-dataset/train_human_generated.pkl", "rb") as f:
    human_generated = pickle.load(f)
    print('Human generated dataset size:', len(human_generated))

human_generated
```

```
  'In the year 1985-86,  purvarti kalyan mantralay ko department of women and child development aur department of welfare mein vibhajit kiya tha.\n',
  'Varsh 1985-86 mein erstwhile Ministry of Welfare was bifurcated into mahila aevam baal vikaas vibhaag and kalyan vibhaag.\n',
  'varsh 1985-86 me erstwhile ministry of welfare ko department of Women and children development and department of welfare me bifurcate kiya gaya tha.\n',
  'In the year 1985-86, poorvavarti kalyan mantralay ko mahila aivam baal vikas vibhag and kalyan vibhag me bifurcate kiya.\n',
  'In the year 1985-86, poorvavarti kalyan mantralay was bifurcated into mahila aivam baal vikas vibhag and kalyan vibhag me vibhakt kiya gaya tha.\n'],
 "A pangolin 's territory is around 3 square miles 8 sq km.\n": ["A pangolin 's territory is around 8 varga kilometer ka hota he.\n",
  'Pangolin ka kshetra is around 3 square miles 8 sq km.\n'],
 'Till, when the Messengers despaired, deeming they were counted liars, Our help came to them and whosoever We willed was delivered. Our might will never be turned back from the people of the sinners.\n': ['Till, When the messengers despaired, un logo ne samaj liya ki vah juthlaye gaye Whosoever we willed was delivered to jise humne chaha najat di aur hamara ajab sinners people ke sar se to tala nahi jata.\n',
  'Pahle ke messengers ne tablige risalat yaha tak ki jab paigamber despaired ho gaye aur deeming they were counted liars, our help came to them and unke pas hamari madad pahuchi. Our might will never be turned back from the people of the sinners.\n'],
 'The Ministry of Labour &amp;amp; Employment, Government of India, instituted the National Safety Awards (Mines) in 1983 for the contest year 1982 to promote a competitive spirit amongst mine operators for the betterment of safety standards in mines and to give due recognition to outstanding safety performance at the national level.\n': ['Ministry of Labour and Employment, Government of India, ne mines me safety standards ke betterment aur national level par outstanding safety performance ko recognition dene ke liye pratyogita varsh 1982 ke liye 1983 me National Safety Awards (Mines) ko institute kiya tha.\n',
  'Mines me safety standards ke betterment aur national level par outstanding safety performance ko recognition dene ke liye the Ministry of Labour &amp;amp; Employment, Government of India, instituted the National Safety Awards (Mines) in 1983 for the contest year 1982.\n',
  'Government of India ke The ministry of labour  & amp; Employment  ne mine operator ki safety standard ke liye aur  competitive spirit ko promote karne ke liye national level par pratiyogita varsh 1983 me outstanding safety performance ka the National Safety Awards(Mines) ka aarambh kiya.\n',
  'To promote a competitive spirit amongst mine operators for the betterment of safety standards in mines and to give due recognition to outstanding safety performance at the national level, bharat Sarkar ke Shram aur Rojgar Mantralay ne National Safety Awards(Mines) ka 1983 me contest year ke liye aarambh kiya. \n'],
 'Dhanpat Rai Shreevastav, who wrote with pen name Premchand (31 July 1880 - 8 October 1936) was one of the greatest Indian , Hindi and Urdu writers.\n': ['Dhanpat Rai Shreevastav, who wrote with pen name Premchand (31 July 1880 - 8 October 1936), Hindi aur Urdu ke mahantam Indian lekhako me se ek he.\n',
  'Premchand (31 july 1880-8 october 1936) ke upnaam se likhne vaale Dhanpat Rai Shreevastav was  one of the greatest Indian, Hindi and Urdu writers.\n'],
 "In the twenty first century Maoist 's rebel spread a lot.\n": ['In the twenty first century Nepal me maoivaadio ka aandolan tez hota gaya.\n',
  "Ikkisvi sadi ki suruaat me, Maoist 's rebel spread a lot.\n"],
 'These poems lack the sombre grandeur of Prantik, for the memory of the Borderland and the haunting sense of the terror and beauty that Meanwhile he was content to relax and watch and savour what simple delight of sight and sound were still left for him on earth.\n': ['These poems lack the sombre grandeur of Prantik, for the memory of the Borderland and the haunting sense of the terror and beauty that is samay ve unke liye earth par bache simple delight of sight and sound ko dekhkar relax se aashvat ho rahe the aur unka aanand uthate prasann the.\n',
  'In Poems sankalano me?Prantik?ki  sombre grandeur nahi thi. Kyonki ab bordarland ki memory ya santras aur beauty ka parivesh tham chala tha aur kavii ek bar fir dharti ki narm baho me tha.Meanwhile he was content to relax and watch and savour what simple delight of sight and sound were still left for him on earth.\n'],
 'He said, “Our Lord is He who gave everything its existence, then guided it. ”\n': ['He said, "hamara Lord wah hai jisne har chiz ko uske(munashib) shurat ataa farmai."\n',
  'Musha ne kaha "Our Lord is He jisne har chiz ko uske shurat ataa farmai"\n'],
 'It was Senator Obama when they created it. They changed the name later.\n': ['It was Senator Obama when they created it. Bad me unhone name badal diya.\n',
  'Jab unhone ise banaya to vah SEnator Obama tha. They changed the name later.\n'],
 ...}
```


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

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

```
1 def preprocess_hin(text):
2     '''Function to preprocess Marathi sentence'''
3     text = re.sub("'", '', text) # remove the quotation marks if any
4     text = ''.join(ch for ch in text if ch not in exclude)
5     text = re.sub("[२३०८९५७४६१]", "", 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> माड्यूल एक संचिका होती है जिसमें या तो...
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> और हमने क़ौमे समूद के पास उनके भाई साल...
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> और जब उन्हें याद दिलाया जाता है तो वे ...
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> तुम्हें का टेड प्राइज़ मिल गया है <end>
	He gone to	उन्होंने बाद	unhonne bad			<start> he gone to	<start> उन्होंने

```
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
```

```
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)
```

```
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
```

```
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_layer
        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))
```

```
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_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')
```

```
Epoch 96 Batch 0 Loss 0.0008
Epoch 96 Loss 0.0111
Time taken for 1 epoch 7.93 sec

Epoch 97 Batch 0 Loss 0.0160
Epoch 97 Loss 0.0155
Time taken for 1 epoch 7.20 sec

Epoch 98 Batch 0 Loss 0.0259
Epoch 98 Loss 0.0537
Time taken for 1 epoch 7.94 sec

Epoch 99 Batch 0 Loss 0.0447
Epoch 99 Loss 0.1171
Time taken for 1 epoch 7.19 sec

Epoch 100 Batch 0 Loss 0.1225
Epoch 100 Loss 0.1011
Time taken for 1 epoch 7.96 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 :  आपकी मांग को बदलने का गाना मंजूर किया हुआ <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 :  और अगर मालूम हो <end>
```

OpenInAPP sentences output

```
1 input_sentence='So even if its a big video I will clearly mention all the products '
2 print('Input sentence in english : ',input_sentence)
3 predicted_output,attention_plot=evaluate(input_sentence)
4 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 :  तो सच है <end>
```

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

```
Input sentence in english :  I was waiting for my bag
Predicted sentence in hindi :  मैंने मैं धैर्य न छूट जाए। <end>
```

```
1
```

```
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)
```

```
Input sentence in english :  definitely share your feedback in the comment section
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-66-854b97a97912> in <cell line: 3>()
      1 input_sentence='definitely share your feedback in the comment section '
      2 print('Input sentence in english : ',input_sentence)
```

✓ 2s completed at 9:42 PM