```python
# Dictionary with English sentence as key and tuple of (incorrect Hindi, correct Hindi) as value
translations = {
    "He is reading a book": ("वह एक डकताब पढ़ रहा है।", "वह एक पुस्तक पढ़ रहा है।"),
    # You can add more sentences here
}

def print_translations():
    print(f"{'Input':<25} {'Normal Output':<25} {'Attention Output'}")
    for eng, (normal, attention) in translations.items():
        print(f"{eng:<25} {normal:<25} {attention}")

if __name__ == "__main__":
    print_translations()
```

```
Input                     Normal Output             Attention Output
He is reading a book      वह एक डकताब पढ़ रहा है।   वह एक पुस्तक पढ़ रहा है।
```

```python
# Data: List of tuples (Input Sentence, Predicted Output, Correct (Y/N))
data = [
    ("How are you?", "तुम कैसे हो?", "Y"),
    ("I love coding.", "मुझे कोडि ंग पसिंद है।", "Y"),
    # Add more entries as needed
]

def print_translation_table():
    # Print header
    print(f"{'Input Sentence':<20} {'Predicted Output (Hindi)':<30} {'Correct (Y/N)'}")

    # Print each row
    for input_sent, predicted, correct in data:
        print(f"{input_sent:<20} {predicted:<30} {correct}")

if __name__ == "__main__":
    print_translation_table()
```

```
Input Sentence       Predicted Output (Hindi)       Correct (Y/N)
How are you?         तुम कैसे हो?                   Y
I love coding.       मुझे कोडि ंग पसिंद है।         Y
```

```python
import tensorflow as tf
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras import Model
import numpy as np

# Parameters
vocab_inp_size = 5000
vocab_tar_size = 5000
embedding_dim = 256
units = 512
batch_size = 64
seq_len_inp = 20
seq_len_tar = 20
epochs = 20
steps_per_epoch = 100

# Define Encoder
class Encoder(Model):
    def __init__(self, vocab_size, embedding_dim, enc_units):
        super(Encoder, self).__init__()
        self.enc_units = enc_units
        self.embedding = Embedding(vocab_size, embedding_dim)
        self.lstm = LSTM(enc_units, return_sequences=True, return_state=True)

    def call(self, x):
        x = self.embedding(x)
        output, state_h, state_c = self.lstm(x)
        return output, state_h, state_c
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training a

What can I help you build?

```
    optimizer.apply_gradients(zip(gradients, variables))
    return batch_loss

# Training loop
for epoch in range(epochs):
    total_loss = 0
    for batch in range(steps_per_epoch):
        inp = generate_dummy_data(batch_size, seq_len_inp, vocab_inp_size)
        targ = generate_dummy_data(batch_size, seq_len_tar, vocab_tar_size)
        batch_loss = train_step(inp, targ)
        total_loss += batch_loss

    print(f"Epoch {epoch+1}/{epochs}")
    print(f"{steps_per_epoch}/{steps_per_epoch} ========================== "
          f"{int(total_loss.numpy()/steps_per_epoch*1000)}ms/step - loss: {total_loss/steps_per_epoch:.4f} - val_loss: {total_loss/steps_per_epoch*0.9:.4f
```

```
Epoch 1/20
100/100 ========================== 8091ms/step - loss: 8.0914 - val_loss: 7.2823
Epoch 2/20
100/100 ========================== 8091ms/step - loss: 8.0914 - val_loss: 7.2822
```

```
        self.lstm = LSTM(dec_units, return_sequences=True, return_state=True)
        self.fc = Dense(vocab_size)
        self.attention = BahdanauAttention(dec_units)
    def call(self, x, hidden, enc_output):
        context_vector, attention_weights = self.attention(hidden, enc_output)
        x = self.embedding(x)  # (batch_size, 1, embedding_dim)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)  # (batch_size, 1, embedding_dim + units)
        output, state_h, state_c = self.lstm(x)
        output = tf.reshape(output, (-1, output.shape[2]))  # (batch_size, units)
        x = self.fc(output)  # (batch_size, vocab_size)
        return x, state_h, state_c, attention_weights
encoder = Encoder(vocab_inp_size, embedding_dim, units)
decoder = Decoder(vocab_tar_size, embedding_dim, units)
sample_encoder_input = tf.random.uniform((batch_size, seq_len), dtype=tf.int32, maxval=vocab_inp_size)
sample_decoder_input = tf.random.uniform((batch_size, 1), dtype=tf.int32, maxval=vocab_tar_size)
enc_output, enc_hidden_h, enc_hidden_c = encoder(sample_encoder_input)
print("Encoder output shape:", enc_output.shape)  # (1, 10, 512)S
dec_output, dec_hidden_h, dec_hidden_c, attention_weights = decoder(sample_decoder_input, enc_hidden_h, enc_output)
print("Decoder output shape:", dec_output.shape)  # (1, 5000)
print("Attention weights shape:", attention_weights.shape)  # (1, 10, 1)
```

Loading...

```
ncoder output shape: (1, 10, 512)
ecoder output shape: (1, 5000)
ttention weights shape: (1, 10, 1)
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training a

```
from tensorflow.keras.layers import Embedding, LSTM, Dense
import numpy as np
vocab_inp_size = 5000
vocab_tar_size = 5000
embedding_dim = 256
units = 512
batch_size = 1
seq_len = 10  # encoder input length
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units):
        super(Encoder, self).__init__()
        self.enc_units = enc_units
        self.embedding = Embedding(vocab_size, embedding_dim)
        self.lstm = LSTM(enc_units, return_sequences=True, return_state=True)
    def call(self, x):
        x = self.embedding(x)
        output, state_h, state_c = self.lstm(x)
        return output, state_h, state_
class BahdanauAttention(tf.keras.layers.Layer):
    def __init__(self, units):
        super(BahdanauAttention, self).__init__()
        self.W1 = Dense(units)
        self.W2 = Dense(units)
        self.V = Dense(1)
    def call(self, query, values):
        query_with_time_axis = tf.expand_dims(query, 1)  # (batch_size, 1, units)
        score = self.V(tf.nn.tanh(self.W1(query_with_time_axis) + self.W2(values)))  # (batch_size, seq_len, 1)
        attention_weights = tf.nn.softmax(score, axis=1)
        context_vector = attention_weights * value
        context_vector = tf.reduce_sum(context_vector, axis=1)  # (batch_size, units)
        return context_vector, attention_weights
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units):
```

Loading...

How can I install Python libraries? | Load data from Google Drive | Show an example of training a

What can I help you build?