NMT PROJECT

English to Hindi Translation

Dhruv Singhal || 500075346 || R177219074 || AIML || Sem 5

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
```

In [2]:

```
import numpy as np
import pandas as pd
import tensorflow
import keras
from keras.models import Model
from keras.layers import Input, LSTM, Dense,TimeDistributed,Embedding,Bidirectional
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from string import digits
import nltk
import re
import string
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', -1)
```

```
Using TensorFlow backend.
C:\Users\Dhruv Singhal\anaconda3\lib\site-packages\ipykernel_launcher.py:1
6: FutureWarning: Passing a negative integer is deprecated in version 1.0
and will not be supported in future version. Instead, use None to not limi
t the column width.
  app.launch_new_instance()
```

In [3]:

```
lines = pd.read_csv('hindi_english_parallel.csv')
lines=lines[:100000]
lines.head()
```

Out[3]:

	hindi	english
0	अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें	Give your application an accessibility workout
1	एक्सेर्साइसर पहुंचनीयता अन्वेषक	Accerciser Accessibility Explorer
2	निचले पटल के लिए डिफोल्ट प्लग-इन खाका	The default plugin layout for the bottom panel
3	ऊपरी पटल के लिए डिफोल्ट प्लग-इन खाका	The default plugin layout for the top panel
4	उन प्लग-इनों की सूची जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है	A list of plugins that are disabled by default

In [4]:

```
# Lowercase all characters  Also called TrueCasing
lines['english']=lines['english'].apply(lambda x: str(x))
lines['hindi']=lines['hindi'].apply(lambda x: str(x))
lines['english']=lines['english'].apply(lambda x: x.lower())
lines['hindi']=lines['hindi'].apply(lambda x: x.lower())
```

In [5]:

```
# Import stopwords with nltk.
from nltk.corpus import stopwords
stop = stopwords.words('english')

lines['english']=lines['english'].apply(lambda x: ' '.join([word for word in x.split() i
lines['hindi']=lines['hindi'].apply(lambda x: ' '.join([word for word in x.split() if wo
```

In [6]:

```
lines['hindi'][0]
```

Out[6]:

```
'अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें'
```

In [7]:

```
# Remove quotes
lines['english']=lines['english'].apply(lambda x: re.sub("'", '', x))
lines['hindi']=lines['hindi'].apply(lambda x: re.sub("'", '', x))
```

In [8]:

```
lines.head()
```

Out[8]:

	hindi	english
0	अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें	give application accessibility workout
1	एक्सेर्साइसर पहुंचनीयता अन्वेषक	accerciser accessibility explorer
2	निचले पटल के लिए डिफोल्ट प्लग-इन खाका	default plugin layout bottom panel
3	ऊपरी पटल के लिए डिफोल्ट प्लग-इन खाका	default plugin layout top panel
4	उन प्लग-इनों की सूची जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है	list plugins disabled default

In [9]:

```
exclude = set(string.punctuation) # Set of all special characters
# Remove all the special characters
lines['english']=lines['english'].apply(lambda x: ''.join(ch for ch in x if ch not in ex
lines['hindi']=lines['hindi'].apply(lambda x: ''.join(ch for ch in x if ch not in excluc
```

In [10]:

```
lines.head()
```

Out[10]:

	hindi	english
0	अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें	give application accessibility workout
1	एक्सेर्साइसर पहुंचनीयता अन्वेषक	accerciser accessibility explorer
2	निचले पटल के लिए डिफोल्ट प्लगइन खाका	default plugin layout bottom panel
3	ऊपरी पटल के लिए डिफोल्ट प्लगइन खाका	default plugin layout top panel
4	उन प्लगइनों की सूची जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है	list plugins disabled default

In [11]:

```
remove_digits = str.maketrans('', '', digits)
```

In [12]:

```
# Remove all numbers from text
remove_digits = str.maketrans('', '', digits)
lines['english']=lines['english'].apply(lambda x: x.translate(remove_digits))
lines['hindi']=lines['hindi'].apply(lambda x: x.translate(remove_digits))

lines['hindi'] = lines['hindi'].apply(lambda x: re.sub("[२३०८१५७९४६]", "", x))

# Remove extra spaces
lines['english']=lines['english'].apply(lambda x: x.strip())
lines['hindi']=lines['hindi'].apply(lambda x: x.strip())
lines['english']=lines['english'].apply(lambda x: re.sub(" +", " ", x))
lines['hindi']=lines['hindi'].apply(lambda x: re.sub(" +", " ", x))
```

In [13]:

```
'hello! how are you buddy?'.strip()
```

Out[13]:

```
'hello! how are you buddy?'
```

In [14]:

```
lines['english'][0]
```

Out[14]:

```
'give application accessibility workout'
```

In [15]:

```
# Add start and end tokens to target sequences
lines['hindi'] = lines['hindi'].apply(lambda x : 'START_ '+ x + ' _END')
```

In [16]:

```
lines['hindi'][0]
```

Out[16]:

```
'START_ अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें _END'
```

In [17]:

```
### Get English and Hindi Vocabulary
all_eng_words=set()
for eng in lines['english']:
    for word in eng.split():
        if word not in all_eng_words:
            all_eng_words.add(word)

all_hindi_words=set()
for hin in lines['hindi']:
    for word in hin.split():
        if word not in all_hindi_words:
            all_hindi_words.add(word)
```

In [18]:

```
lines.head()
```

Out[18]:

	hindi	english
0	START_ अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें _END	give application accessibility workout
1	START_ एक्सेसर्साइसर पहुंचनीयता अन्वेषक _END	accerciser accessibility explorer
2	START_ निचले पटल के लिए डिफोल्ट प्लगइन खाका _END	default plugin layout bottom panel
3	START_ ऊपरी पटल के लिए डिफोल्ट प्लगइन खाका _END	default plugin layout top panel
4	START_ उन प्लगइनों की सूची जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है _END	list plugins disabled default

In [19]:

```
lines['length_eng']=lines['english'].apply(lambda x:len(x.split(" ")))
lines['length_hin']=lines['hindi'].apply(lambda x:len(x.split(" ")))
```

In [20]:

```
lines.head()
lines[lines['length_eng']>30].shape
```

Out[20]:

```
(103, 4)
```

In [21]:

```
lines=lines[lines['length_eng']<=20]
lines=lines[lines['length_hin']<=20]
```

In [22]:

```
print("maximum length of Hindi Sentence ",max(lines['length_hin']))
print("maximum length of English Sentence ",max(lines['length_eng']))
```

```
maximum length of Hindi Sentence  20
maximum length of English Sentence  19
```

In [23]:

```
max_length_src=max(lines['length_hin'])
max_length_tar=max(lines['length_eng'])
```

In [24]:

```
input_words = sorted(list(all_eng_words))
target_words = sorted(list(all_hindi_words))
num_encoder_tokens = len(all_eng_words)
num_decoder_tokens = len(all_hindi_words)
num_encoder_tokens, num_decoder_tokens
```

Out[24]:

```
(5692, 8427)
```

In [25]:

```
num_decoder_tokens
```

Out[25]:

```
8427
```

In [26]:

```
num_decoder_tokens += 1
```

In [27]:

```
num_decoder_tokens
```

Out[27]:

```
8428
```

In [28]:

```
input_token_index = dict([(word, i+1) for i, word in enumerate(input_words)])
target_token_index = dict([(word, i+1) for i, word in enumerate(target_words)])
```

In [29]:

```
input_token_index
```

```
 'anyway': 221,
 'anywhere': 222,
 'ap': 223,
 'apache': 224,
 'aperture': 225,
 'api': 226,
 'apia': 227,
 'apop': 228,
 'app': 229,
 'apparently': 230,
 'appear': 231,
 'appearance': 232,
 'appearing': 233,
 'appears': 234,
 'append': 235,
 'appending': 236,
 'apple': 237,
 'applet': 238,

 'application': 239,
```

In [30]:

```
reverse_input_char_index = dict((i, word) for word, i in input_token_index.items())
reverse_target_char_index = dict((i, word) for word, i in target_token_index.items())
```

In [31]:

```
reverse_input_char_index
```

Out[31]:

```
{1: 'a',
 2: 'aaiun',
 3: 'ababa',
 4: 'abbreviations',
 5: 'abbrevweekdayname',
 6: 'abc',
 7: 'abcdefghijk',
 8: 'abidjan',
 9: 'ability',
 10: 'able',
 11: 'abnormal',
 12: 'abort',
 13: 'aborted',
 14: 'aborting',
 15: 'about',
 16: 'about…',
 17: 'above',
 18: 'absolute',
```

In [32]:

```
lines.head(10)
```

Out[32]:

	hindi	english	length_eng	length_hin
0	START_ अपने अनुप्रयोग को पहुंचनीयता व्यायाम का लाभ दें _END	give application accessibility workout	4	10
1	START_ एक्सेर्साइसर पहुंचनीयता अन्वेषक _END	accerciser accessibility explorer	3	5
2	START_ निचले पटल के लिए डिफोल्ट प्लगइन खाका _END	default plugin layout bottom panel	5	9
3	START_ ऊपरी पटल के लिए डिफोल्ट प्लगइन खाका _END	default plugin layout top panel	5	9
4	START_ उन प्लगइनों की सूची जिन्हें डिफोल्ट रूप से निष्क्रिय किया गया है _END	list plugins disabled default	4	14
5	START_ अवधि को हाइलाइट रकें _END	highlight duration	2	6
6	START_ पहुंचनीय आसंधि नोड को चुनते समय हाइलाइट बक्से की अवधि _END	duration highlight box selecting accessible nodes	6	12
7	START_ सीमांत बोर्डर के रंग को हाइलाइट करें _END	highlight border color	3	9
8	START_ हाइलाइट किए गए सीमांत का रंग और अपारदर्शिता। _END	color opacity highlight border	4	10
9	START_ भराई के रंग को हाइलाइट करें _END	highlight fill color	3	8

In [33]:

```
from sklearn.model_selection import train_test_split
X, y = lines['english'], lines['hindi']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,random_state=
X_train.shape, X_test.shape
```

Out[33]:

```
((78042,), (19511,))
```

In [34]:

```
X_train
```

Out[34]:

```
81104    server response match
96026    america el salvador
30434    simulation video disc burning
75110    options
93620    notify new messages inbox only
                        ...
6328     search help
55958    failed retrieve personal information server
78506    pkcs sha rsa encryption
864      lo gin helper
15948    new library…
Name: english, Length: 78042, dtype: object
```

In [35]:

```
encoder_input_data = np.zeros((2, max_length_src),dtype='float32')
decoder_input_data = np.zeros((2, max_length_tar),dtype='float32')
decoder_target_data = np.zeros((2, max_length_tar, num_decoder_tokens),dtype='float32')
```

In [36]:

```
def generate_batch(X = X_train, y = y_train, batch_size = 128):
    ''' Generate a batch of data '''
    while True:
        for j in range(0, len(X), batch_size):
            encoder_input_data = np.zeros((batch_size, max_length_src),dtype='float32')
            decoder_input_data = np.zeros((batch_size, max_length_tar),dtype='float32')
            decoder_target_data = np.zeros((batch_size, max_length_tar, num_decoder_toke
            for i, (input_text, target_text) in enumerate(zip(X[j:j+batch_size], y[j:j+b
                for t, word in enumerate(input_text.split()):
                    encoder_input_data[i, t] = input_token_index[word] # encoder input s
                for t, word in enumerate(target_text.split()):
                    if t<len(target_text.split())-1:
                        decoder_input_data[i, t] = target_token_index[word] # decoder in
                    if t>0:
                        # decoder target sequence (one hot encoded)
                        # does not include the START_ token
                        # Offset by one timestep
                        decoder_target_data[i, t - 1, target_token_index[word]] = 1.
            yield([encoder_input_data, decoder_input_data], decoder_target_data)
```

In [37]:

```
latent_dim = 300
# Encoder
encoder_inputs = Input(shape=(None,))
enc_emb =  Embedding(num_encoder_tokens+1, latent_dim, mask_zero = True)(encoder_inputs
encoder_lstm = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(enc_emb)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

```
WARNING:tensorflow:From C:\Users\Dhruv Singhal\AppData\Roaming\Python\Pyth
on37\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:163
0: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resou
rce_variable_ops) with constraint is deprecated and will be removed in a f
uture version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From C:\Users\Dhruv Singhal\AppData\Roaming\Python\Pyth
on37\site-packages\tensorflow_core\python\keras\backend.py:3994: where (fr
om tensorflow.python.ops.array_ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [38]:

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(num_decoder_tokens+1, latent_dim, mask_zero = True)
dec_emb = dec_emb_layer(decoder_inputs)
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(dec_emb,
                                     initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

In [39]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

In [40]:

```
model.summary()
train_samples = len(X_train)
val_samples = len(X_test)
batch_size = 128
epochs = 10
```

```
Model: "model_1"
__________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================
========================
input_1 (InputLayer)            (None, None)         0

__________________________________________________________________________
________________________
input_2 (InputLayer)            (None, None)         0

__________________________________________________________________________
________________________
embedding_1 (Embedding)         (None, None, 300)    1707900     input_1
[0][0]

__________________________________________________________________________
________________________
embedding_2 (Embedding)         (None, None, 300)    2528700     input_2
[0][0]

__________________________________________________________________________
________________________
lstm_1 (LSTM)                   [(None, 300), (None, 721200      embedding
_1[0][0]

__________________________________________________________________________
________________________
lstm_2 (LSTM)                   [(None, None, 300),  721200      embedding
_2[0][0]
                                                                 lstm_1[0]
[1]
                                                                 lstm_1[0]
[2]
__________________________________________________________________________
________________________
dense_1 (Dense)                 (None, None, 8428)   2536828     lstm_2[0]
[0]
==========================================================================
========================
Total params: 8,215,828
Trainable params: 8,215,828
Non-trainable params: 0
__________________________________________________________________________
________________________
```

In [41]:

```
a, b = next(generate_batch())
```

In [42]:

```
model.fit_generator(generator = generate_batch(X_train, y_train, batch_size = batch_size
                    steps_per_epoch = train_samples/batch_size,
                    epochs=10,
                    validation_data = generate_batch(X_test, y_test, batch_size = batch_
                    validation_steps = val_samples/batch_size)
```

```
WARNING:tensorflow:From C:\Users\Dhruv Singhal\anaconda3\lib\site-packages
\keras\backend\tensorflow_backend.py:422: The name tf.global_variables is
deprecated. Please use tf.compat.v1.global_variables instead.

Epoch 1/10
610/609 [==============================] - 817s 1s/step - loss: 1.4466 - a
ccuracy: 0.2362 - val_loss: 0.4933 - val_accuracy: 0.3042
Epoch 2/10
610/609 [==============================] - 842s 1s/step - loss: 0.9675 - a
ccuracy: 0.3897 - val_loss: 0.3175 - val_accuracy: 0.4828
Epoch 3/10
610/609 [==============================] - 873s 1s/step - loss: 0.6032 - a
ccuracy: 0.5744 - val_loss: 0.2032 - val_accuracy: 0.6457
Epoch 4/10
610/609 [==============================] - 855s 1s/step - loss: 0.3855 - a
ccuracy: 0.7197 - val_loss: 0.1386 - val_accuracy: 0.7503
Epoch 5/10
610/609 [==============================] - 859s 1s/step - loss: 0.2630 - a
ccuracy: 0.8090 - val_loss: 0.0977 - val_accuracy: 0.8155
Epoch 6/10
610/609 [==============================] - 853s 1s/step - loss: 0.1894 - a
ccuracy: 0.8628 - val_loss: 0.0729 - val_accuracy: 0.8554
Epoch 7/10
610/609 [==============================] - 817s 1s/step - loss: 0.1427 - a
ccuracy: 0.8954 - val_loss: 0.0561 - val_accuracy: 0.8774
Epoch 8/10
610/609 [==============================] - 818s 1s/step - loss: 0.1119 - a
ccuracy: 0.9162 - val_loss: 0.0452 - val_accuracy: 0.8937
Epoch 9/10
610/609 [==============================] - 2227s 4s/step - loss: 0.0907 -
accuracy: 0.9300 - val_loss: 0.0361 - val_accuracy: 0.9029
Epoch 10/10
610/609 [==============================] - 940s 2s/step - loss: 0.0765 - a
ccuracy: 0.9388 - val_loss: 0.0334 - val_accuracy: 0.9087
```

Out[42]:

```
<keras.callbacks.callbacks.History at 0x271afb40288>
```

In [43]:

```
train_gen = generate_batch(X_train, y_train, batch_size = 1)
k=-1
```

In [44]:

```
# Encode the input sequence to get the "thought vectors"
encoder_model = Model(encoder_inputs, encoder_states)

# Decoder setup
# Below tensors will hold the states of the previous time step
decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]

dec_emb2= dec_emb_layer(decoder_inputs) # Get the embeddings of the decoder sequence

# To predict the next word in the sequence, set the initial states to the states from th
decoder_outputs2, state_h2, state_c2 = decoder_lstm(dec_emb2, initial_state=decoder_stat
decoder_states2 = [state_h2, state_c2]
decoder_outputs2 = decoder_dense(decoder_outputs2) # A dense softmax layer to generate p

# Final decoder model
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs2] + decoder_states2)
```

In [45]:

```python
def decode_sequence(input_seq):
    # Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)
    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))
    # Populate the first character of target sequence with the start character.
    target_seq[0, 0] = target_token_index['START_']

    # Sampling loop for a batch of sequences
    # (to simplify, here we assume a batch of size 1).
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = reverse_target_char_index[sampled_token_index]
        decoded_sentence += ' '+sampled_char

        # Exit condition: either hit max length
        # or find stop character.
        if (sampled_char == '_END' or
           len(decoded_sentence) > 50):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index

        # Update states
        states_value = [h, c]

    return decoded_sentence
```

In [55]:

```python
from nltk.translate.bleu_score import sentence_bleu
k+=1
ref = [ y_train[k:k+1].values[0][6:-4]]
test = decoded_sentence[:-4]
print('BLEU score for test-> {}'.format(sentence_bleu(ref, test)))
```

```
BLEU score for test-> 0.45507751084172515
```

In [ ]:

```python

```