

```

1 import pandas as pd
2 import re
3 import tensorflow as tf
4 from tensorflow.keras.layers import Embedding, LSTM, Dense
5 from tensorflow.keras.models import Model
6 from tensorflow.keras.preprocessing.text import Tokenizer
7 from tensorflow.keras.preprocessing.sequence import pad_sequences
8 import numpy as np
9 import nltk.translate.bleu_score as bleu
10 import random
11 import string
12 from sklearn.model_selection import train_test_split
13 import os
14 import time

```

```

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

```

Mounted at /content/drive

```

1 eng_hin=pd.read_csv('/content/drive/MyDrive/PRL/task/data/synthetic-dataset/train.csv')
2 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) कौमे 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 नहीं...	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

```

1 # create a new dataframe of english and hinglish column
2 df = pd.DataFrame()
3 df["english"] = eng_hin["English"]
4 df["hindi"] = eng_hin["Hindi"]
5 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) कौमे 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 नहीं...
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...

```

1 eng_hin.dropna(inplace=True)
2 eng_hin.shape

```

(2766, 7)

```

1 import pickle
2
3 with open("/content/drive/MyDrive/PRL/task/data/human-generated-dataset/train_human_generated.pkl", "rb") as f:
4     human_generated = pickle.load(f)
5     print('Human generated dataset size:', len(human_generated))
6
7 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 & 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 & 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 & 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 jisme har chiz ko uske(munashib) shurat ataa farmai."\n',

'Musha ne kaha "Our Lord is He jisme 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("[२३०८९५७८९४५]", "", text) # remove the digits
```

```

6 text = text.strip()
7 text = re.sub(" +", " ", text) # remove extra spaces
8 text = '<start> ' + text + ' <end>'
9 return text

1 eng_hin['english'] = df['english'].apply(preprocess)
2 eng_hin['hindi'] = df['hindi'].apply(preprocess_hin)
3
4 eng_hin.rename(columns={"english": "english", "hindi": "hindi"},inplace=True)
5
6 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) क्रौमे 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 क्रौमे समूद के पास उनके भाई साल...
2	and, when reminded, do not remember	और जब उन्हें याद दिलाया जाता है, तो वे याद नहीं...	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.	तुम्हें २०११ का टेड प्राइज़ मिल गया है.	tumhen २०११ ka ted prize mil gaya hai	9	1	<start> you won the ted prize <end>	<start> तुम्हें का टेड प्राइज़ मिल गया है <end>
	He gone to	उन्होंने बाद	unhone bad			<start> he gone to	<start> उन्होंने

```

1 def tokenize(lang):
2
3     lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
4     lang_tokenizer.fit_on_texts(lang)
5
6     tensor = lang_tokenizer.texts_to_sequences(lang)
7
8     tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor,padding='post',maxlen=20,dtype='int32')
9
10    return tensor, lang_tokenizer

1 def load_dataset():
2
3     input_tensor, inp_lang_tokenizer = tokenize(eng_hin['english'].values)
4     target_tensor, targ_lang_tokenizer = tokenize(eng_hin['hindi'].values)
5
6     return input_tensor, target_tensor, inp_lang_tokenizer, targ_lang_tokenizer

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

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

1 input_tensor_train, input_tensor_val, target_tensor_train, target_tensor_val = train_test_split(input_tensor, target_tei
2
3 print(len(input_tensor_train), len(target_tensor_train), len(input_tensor_val), len(target_tensor_val))

2212 2212 554 554

1 BUFFER_SIZE = len(input_tensor_train)
2 BATCH_SIZE = 32
3 N_BATCH = BUFFER_SIZE//BATCH_SIZE
4 embedding_dim = 256
5 units = 1024
6 steps_per_epoch = len(input_tensor_train)//BATCH_SIZE
7
8 vocab_inp_size =len(inp_lang.word_index.keys())
9 vocab_tar_size =len(targ_lang.word_index.keys())

```

```

10
11 dataset = tf.data.Dataset.from_tensor_slices((input_tensor_train, target_tensor_train)).shuffle(BUFFER_SIZE)
12 dataset = dataset.batch(BATCH_SIZE).drop_remainder=True)

1 embeddings_index = dict()
2 f = open('/content/drive/MyDrive/PRL/task/data/synthetic-dataset/glove.6B.300d.txt')
3 for line in f:
4     values = line.split()
5     word = values[0]
6     coefs = np.asarray(values[1:], dtype='float32')
7     embeddings_index[word] = coefs
8 f.close()
9
10 embedding_matrix = np.zeros((vocab_inp_size+1, 300))
11 for word, i in inp_lang.word_index.items():
12     embedding_vector = embeddings_index.get(word)
13     if embedding_vector is not None:
14         embedding_matrix[i] = embedding_vector

1 class Encoder(tf.keras.Model):
2     def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
3         super(Encoder, self).__init__()
4         self.batch_sz = batch_sz
5         self.enc_units = enc_units
6         self.embedding = tf.keras.layers.Embedding(input_dim=vocab_size, output_dim=embedding_dim, name="embedding_layer")
7         self.gru = tf.keras.layers.GRU(units, return_sequences=True, return_state=True, recurrent_activation='sigmoid',
8
9     def call(self, x, hidden):
10         x = self.embedding(x)
11         output, state = self.gru(x, initial_state = hidden)
12         return output, state
13
14     def initialize_hidden_state(self):
15         return tf.zeros((self.batch_sz, self.enc_units))

```

```

1 class Decoder(tf.keras.Model):
2     def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
3         super(Decoder, self).__init__()
4         self.batch_sz = batch_sz
5         self.dec_units = dec_units
6         self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
7         self.gru = tf.keras.layers.GRU(units, return_sequences=True, return_state=True, recurrent_activation='sigmoid',
8         self.fc = tf.keras.layers.Dense(vocab_size)
9
10        # used for attention
11        self.W1 = tf.keras.layers.Dense(self.dec_units)
12        self.W2 = tf.keras.layers.Dense(self.dec_units)
13        self.V = tf.keras.layers.Dense(1)
14
15    def call(self, x, hidden, enc_output):
16
17        hidden_with_time_axis = tf.expand_dims(hidden, 1)
18
19        score = self.V(tf.nn.tanh(self.W1(enc_output) + self.W2(hidden_with_time_axis)))
20
21        attention_weights = tf.nn.softmax(score, axis=1)
22
23        context_vector = attention_weights * enc_output
24        context_vector = tf.reduce_sum(context_vector, axis=1)
25
26        x = self.embedding(x)
27
28        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
29
30        output, state = self.gru(x)
31
32        output = tf.reshape(output, (-1, output.shape[2]))
33
34        x = self.fc(output)
35
36        return x, state, attention_weights
37
38    def initialize_hidden_state(self):
39        return tf.zeros((self.batch_sz, self.dec_units))
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```
8     dec_hidden = enc_hidden
9
10    dec_input = tf.expand_dims([targ_lang.word_index['<start>']] * BATCH_SIZE, 1)
11
12    for t in range(1, targ.shape[1]):
13        predictions, dec_hidden, _ = decoder(dec_input, dec_hidden, enc_output)
14
15        loss += loss_function(targ[:, t], predictions)
16
17        dec_input = tf.expand_dims(targ[:, t], 1)
18
19    batch_loss = (loss / int(targ.shape[1]))
20
21    variables = encoder.trainable_variables + decoder.trainable_variables
22
23    gradients = tape.gradient(loss, variables)
24
25    optimizer.apply_gradients(zip(gradients, variables))
26
27    return batch_loss
```

```
1 EPOCHS = 100
2
3 for epoch in range(EPOCHS):
4     start = time.time()
5
6     enc_hidden = encoder.initialize_hidden_state()
7     total_loss = 0
8
9     for (batch, (inp, targ)) in enumerate(dataset.take(steps_per_epoch)):
10         batch_loss = train_step(inp, targ, enc_hidden)
11         total_loss += batch_loss
12
13     if batch % 100 == 0:
14         print(f'Epoch {epoch+1} Batch {batch} Loss {batch_loss.numpy():.4f}')
15     if (epoch + 1) % 2 == 0:
16         checkpoint.save(file_prefix=checkpoint_prefix)
17
18     print(f'Epoch {epoch+1} Loss {total_loss/steps_per_epoch:.4f}')
19     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

```
1 def evaluate(sentence):
2     attention_plot = np.zeros((max_length_targ, max_length_inp))
3
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)
9
10    result = ''
11
12    hidden = [tf.zeros((1, units))]
13    enc_out, enc_hidden = encoder(inputs, hidden)
14
15    dec_hidden = enc_hidden
16    dec_input = tf.expand_dims([targ_lang.word_index['<start>']], 0)
17
18    for t in range(max_length_targ):
19        predictions, dec_hidden, attention_weights = decoder(dec_input,
20                                                            dec_hidden,
21                                                            enc_out)
22        # storing the attention weights to plot later on
23        attention_weights = tf.reshape(attention_weights, (-1, ))
24        attention_plot[t] = attention_weights.numpy()
25        predicted_id = tf.argmax(predictions[0]).numpy()
26
27        result += targ_lang.index_word[predicted_id] + ' '
28
29        if targ_lang.index_word[predicted_id] == '<end>':
30            return result, attention_plot
31
32        # the predicted ID is fed back into the model
33        dec_input = tf.expand_dims([predicted_id], 0)
34
35    return result, attention_plot
```

```
1 input_sentence= 'please ensure that you use the appropriate form '
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 : please ensure that you use the appropriate form
 Predicted sentence in hindi : आपकी मांग को बदलने का गाना मंजूर किया हुआ <end>

```
1 input_sentence='and do something with it to change the world '
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 : 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

