In [1]:
```python
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F

import numpy as np
import pandas as pd

import os
import re
import random
import unicodedata
import string

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
MAX_LENGTH = 21
```

```
C:\Users\Admin\.conda\envs\torch\lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgre
ss not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.i
o/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [2]:
```python
SOS_token = 0
EOS_token = 1


class Lang:
    def __init__(self, name):
        self.name = name
        self.word2index = {}
        self.word2count = {}
        self.index2word = {0: "SOS", 1: "EOS"}
        self.n_words = 2  # Count SOS and EOS

    def addSentence(self, sentence):
        for word in sentence.split(' '):
            self.addWord(word)

    def addWord(self, word):
        if word not in self.word2index:
            self.word2index[word] = self.n_words
            self.word2count[word] = 1
            self.index2word[self.n_words] = word
            self.n_words += 1
        else:
            self.word2count[word] += 1
```

In [3]:
```python
# Turn a Unicode string to plain ASCII, thanks to
# https://stackoverflow.com/a/518232/2809427
def unicodeToAscii(s):
    return ''.join(
        c for c in unicodedata.normalize('NFD', s)
        if unicodedata.category(c) != 'Mn'
    )

# Lowercase, trim, and remove non-letter characters


def normalizeString(s):
    #s = unicodeToAscii(s.lower().strip())
    s = s.lower()
    s = re.sub(r"([.!?])", r" \1", s)
    s = re.sub("[0-9]",r" \1",s)
    #s = re.sub(r"[^a-zA-Z.!?]+", r" ", s)
    return s
```

In [4]:
```
def readLangs(lang1, lang2, reverse=False):
    print("Reading lines...")

    # Read the file and split into lines
    #lines = open('data/%s-%s.txt' % (lang1, lang2), encoding='utf-8').\
        #read().strip().split('\n')
    data = pd.read_csv('Hindi_English_Truncated_Corpus.csv',encoding='utf8')
    data = data[[lang1,lang2]]
    # Split every line into pairs and normalize
    #data[lang1] = data[lang1].apply(lambda x: normalizeString(str(x)))#(normalizeString
    #data[lang2] = data[lang2].apply(lambda x: normalizeString(str(x)))#(normalizeString

    to_exclude = set(string.punctuation) # Set of all special characters
    print("punctuations to exclude:: ",to_exclude)
    # Remove all the special characters
    data[lang1]=data[lang1].apply(lambda x: ''.join(ch for ch in str(x) if ch not in to_
    data[lang2]=data[lang2].apply(lambda x: ''.join(ch for ch in str(x) if ch not in to_

    data['pairs'] = [[e for e in row if e==e] for row in data[[lang1,lang2]].values.toli

    pairs = data['pairs'].tolist()

    #Reverse pairs, make Lang instances
    if reverse:
        pairs = [list(reversed(p)) for p in pairs]
        input_lang = Lang(lang2)
        output_lang = Lang(lang1)
    else:
        input_lang = Lang(lang1)
        output_lang = Lang(lang2)

    return input_lang, output_lang, pairs
```

In [5]:
```
def filterPair(p):
    return len(p[0].split(' ')) < MAX_LENGTH and len(p[1].split(' ')) < MAX_LENGTH
```

In [6]:
```
def filterPairs(pairs):
    return [pair for pair in pairs if filterPair(pair)]
```

In [7]:
```
def prepareData(lang1, lang2,reverse=False):
    input_lang, output_lang, pairs = readLangs(lang1, lang2,reverse)

    print("Read %s sentence pairs" % len(pairs))
    #print(pairs)
    pairs = filterPairs(pairs)
    print("Trimmed to %s sentence pairs" % len(pairs))
    print("Counting words...")
    for pair in pairs:
        input_lang.addSentence(pair[0])
        output_lang.addSentence(pair[1])
    print("Counted words:")
    print(input_lang.name, input_lang.n_words)
    print(output_lang.name, output_lang.n_words)
    return input_lang, output_lang, pairs
```

In [8]:
```
input_lang, output_lang, pairs = prepareData('english_sentence', 'hindi_sentence',False)
print(random.choice(pairs))
```

```
Reading lines...
punctuations to exclude::  {'&', '?', '|', ')', '"', '\\', '*', '!', "'", '}', '[', '(',
'~', '>', '=', '%', '{', '#', '_', '+', '.', ':', ',', '-', '$', '@', '^', '`', '/',
'<', ']', ';'}
Read 127607 sentence pairs
Trimmed to 85781 sentence pairs
Counting words...
Counted words:
english_sentence 50965
hindi_sentence 44262
['A selforganizing system is one', 'एक स्व संगठनीय प्रणाली वो होती है']
```

In [9]:
```
class EncoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(input_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size)

    def forward(self, input, hidden):
        embedded = self.embedding(input).view(1, 1, -1)
        output = embedded
        output, hidden = self.gru(output, hidden)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

In [10]:
```
class DecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size):
        super(DecoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(output_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        output = self.embedding(input).view(1, 1, -1)
        output = F.relu(output)
        output, hidden = self.gru(output, hidden)
        output = self.softmax(self.out(output[0]))
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

In [11]:
```
class AttnDecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size, dropout_p=0.1, max_length=MAX_LENGTH):
        super(AttnDecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.dropout_p = dropout_p
        self.max_length = max_length

        self.embedding = nn.Embedding(self.output_size, self.hidden_size)
        self.attn = nn.Linear(self.hidden_size * 2, self.max_length)
        self.attn_combine = nn.Linear(self.hidden_size * 2, self.hidden_size)
        self.dropout = nn.Dropout(self.dropout_p)
        self.gru = nn.GRU(self.hidden_size, self.hidden_size)
        self.out = nn.Linear(self.hidden_size, self.output_size)

    def forward(self, input, hidden, encoder_outputs):
        embedded = self.embedding(input).view(1, 1, -1)
        embedded = self.dropout(embedded)
        attn_weights = F.softmax(
            self.attn(torch.cat((embedded[0], hidden[0]), 1)), dim=1)
        attn_applied = torch.bmm(attn_weights.unsqueeze(0),
                                 encoder_outputs.unsqueeze(0))

        output = torch.cat((embedded[0], attn_applied[0]), 1)
        output = self.attn_combine(output).unsqueeze(0)

        output = F.relu(output)
        output, hidden = self.gru(output, hidden)

        output = F.log_softmax(self.out(output[0]), dim=1)
        return output, hidden, attn_weights

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

In [12]:
```
def indexesFromSentence(lang, sentence):
    return [lang.word2index[word] for word in sentence.split(' ')]


def tensorFromSentence(lang, sentence):
    indexes = indexesFromSentence(lang, sentence)
    indexes.append(EOS_token)
    return torch.tensor(indexes, dtype=torch.long, device=device).view(-1, 1)


def tensorsFromPair(pair):
    input_tensor = tensorFromSentence(input_lang, pair[0])
    target_tensor = tensorFromSentence(output_lang, pair[1])
    return (input_tensor, target_tensor)
```

In [13]:
```
teacher_forcing_ratio = 0.5


def train(input_tensor, target_tensor, encoder, decoder, encoder_optimizer, decoder_opti
    encoder_hidden = encoder.initHidden()

    encoder_optimizer.zero_grad()
    decoder_optimizer.zero_grad()

    input_length = input_tensor.size(0)
    target_length = target_tensor.size(0)

    encoder_outputs = torch.zeros(max_length, encoder.hidden_size, device=device)

    loss = 0

    for ei in range(input_length):
        encoder_output, encoder_hidden = encoder(
            input_tensor[ei], encoder_hidden)
        encoder_outputs[ei] = encoder_output[0, 0]

    decoder_input = torch.tensor([[SOS_token]], device=device)

    decoder_hidden = encoder_hidden

    use_teacher_forcing = True if random.random() < teacher_forcing_ratio else False

    if use_teacher_forcing:
        # Teacher forcing: Feed the target as the next input
        for di in range(target_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            loss += criterion(decoder_output, target_tensor[di])
            decoder_input = target_tensor[di]  # Teacher forcing

    else:
        # Without teacher forcing: use its own predictions as the next input
        for di in range(target_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            topv, topi = decoder_output.topk(1)
            decoder_input = topi.squeeze().detach()  # detach from history as input

            loss += criterion(decoder_output, target_tensor[di])
            if decoder_input.item() == EOS_token:
                break

    loss.backward()

    encoder_optimizer.step()
    decoder_optimizer.step()

    return loss.item() / target_length
```

In [14]:
```
import time
import math


def asMinutes(s):
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)


def timeSince(since, percent):
    now = time.time()
    s = now - since
    es = s / (percent)
    rs = es - s
    return '%s (- %s)' % (asMinutes(s), asMinutes(rs))
```

In [15]:
```
def trainIters(encoder, decoder, n_iters, print_every=1000, plot_every=1000, learning_ra
    start = time.time()
    plot_losses = []
    print_loss_total = 0  # Reset every print_every
    plot_loss_total = 0  # Reset every plot_every

    #encoder_optimizer = optim.SGD(encoder.parameters(), lr=learning_rate)
    #decoder_optimizer = optim.SGD(decoder.parameters(), lr=learning_rate)
    encoder_optimizer = optim.AdamW(encoder.parameters(), lr=learning_rate)
    decoder_optimizer = optim.AdamW(decoder.parameters(), lr=learning_rate)
    training_pairs = [tensorsFromPair(random.choice(pairs))
                      for i in range(n_iters)]
    criterion = nn.CrossEntropyLoss()

    for iter in range(1, n_iters + 1):
        training_pair = training_pairs[iter - 1]
        input_tensor = training_pair[0]
        target_tensor = training_pair[1]

        loss = train(input_tensor, target_tensor, encoder,
                     decoder, encoder_optimizer, decoder_optimizer, criterion)
        print_loss_total += loss
        plot_loss_total += loss

        if iter % print_every == 0:
            print_loss_avg = print_loss_total / print_every
            print_loss_total = 0
            print('%s (%d %d%%) %.4f' % (timeSince(start, iter / n_iters),
                                         iter, iter / n_iters * 100, print_loss_avg))

        if iter % plot_every == 0:
            plot_loss_avg = plot_loss_total / plot_every
            plot_losses.append(plot_loss_avg)
            plot_loss_total = 0

    showPlot(plot_losses)
```

In [16]:
```
import matplotlib.pyplot as plt
#plt.switch_backend('agg')
import matplotlib.ticker as ticker
import numpy as np


def showPlot(points):
    plt.figure()
    fig, ax = plt.subplots()
    # this locator puts ticks at regular intervals
    loc = ticker.MultipleLocator(base=0.2)
    ax.yaxis.set_major_locator(loc)
    plt.plot(points)
```

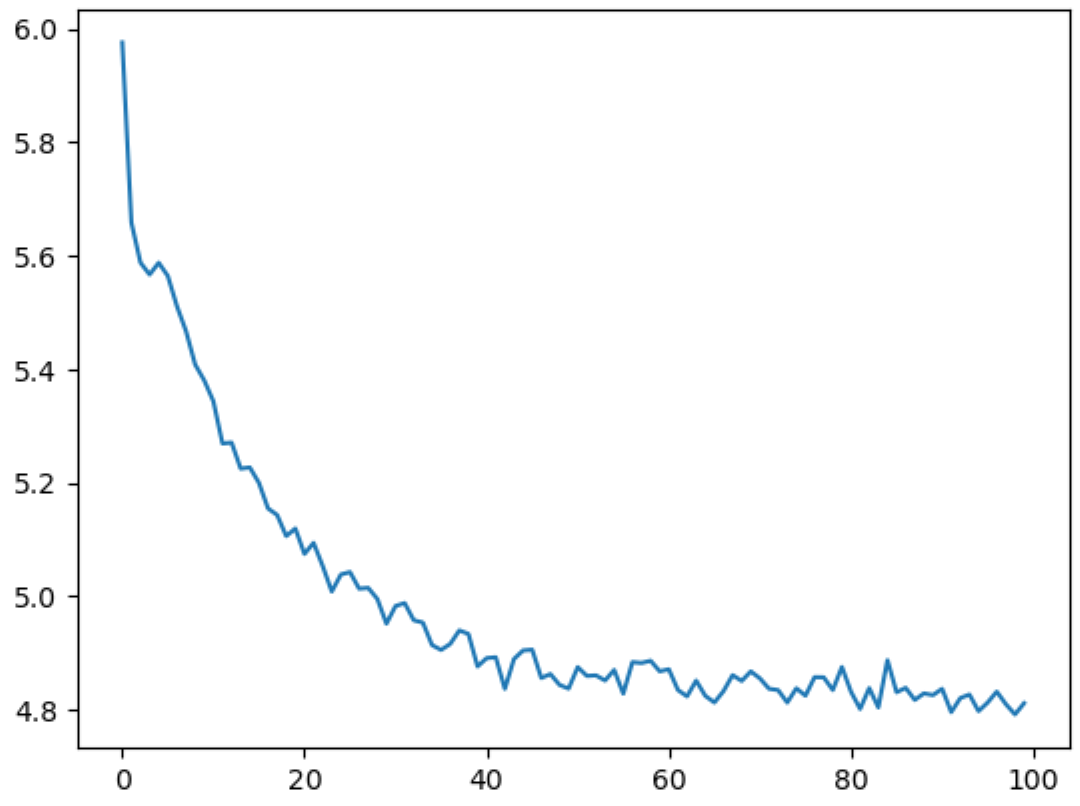
In [17]:
```
def evaluate(encoder, decoder, sentence, max_length=MAX_LENGTH):
    with torch.no_grad():
        input_tensor = tensorFromSentence(input_lang, sentence)
        input_length = input_tensor.size()[0]
        encoder_hidden = encoder.initHidden()

        encoder_outputs = torch.zeros(max_length, encoder.hidden_size, device=device)

        for ei in range(input_length):
            encoder_output, encoder_hidden = encoder(input_tensor[ei],
                                                     encoder_hidden)
            encoder_outputs[ei] += encoder_output[0, 0]

        decoder_input = torch.tensor([[SOS_token]], device=device)  # SOS

        decoder_hidden = encoder_hidden

        decoded_words = []
        decoder_attentions = torch.zeros(max_length, max_length)

        for di in range(max_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            decoder_attentions[di] = decoder_attention.data
            topv, topi = decoder_output.data.topk(1)
            if topi.item() == EOS_token:
                decoded_words.append('<EOS>')
                break
            else:
                decoded_words.append(output_lang.index2word[topi.item()])

            decoder_input = topi.squeeze().detach()

        return decoded_words, decoder_attentions[:di + 1]
```

In [18]:
```
def evaluateRandomly(encoder, decoder, n=10):
    for i in range(n):
        pair = random.choice(pairs)
        print('>', pair[0])
        print('=', pair[1])
        output_words, attentions = evaluate(encoder, decoder, pair[0])
        output_sentence = ' '.join(output_words)
        print('<', output_sentence)
        print('')
```

In [19]:
```
hidden_size = 256
encoder1 = EncoderRNN(input_lang.n_words, hidden_size).to(device)
attn_decoder1 = AttnDecoderRNN(hidden_size, output_lang.n_words, dropout_p=0.2).to(devic

trainIters(encoder1, attn_decoder1, 1000000, print_every=10000,plot_every=10000) #560000
```

```
13m 48s (- 1367m 21s) (10000 1%) 5.9757
25m 15s (- 1237m 35s) (20000 2%) 5.6581
36m 54s (- 1193m 32s) (30000 3%) 5.5876
48m 30s (- 1164m 4s) (40000 4%) 5.5666
60m 13s (- 1144m 16s) (50000 5%) 5.5874
72m 7s (- 1129m 57s) (60000 6%) 5.5641
83m 23s (- 1107m 57s) (70000 7%) 5.5109
94m 11s (- 1083m 6s) (80000 8%) 5.4668
105m 0s (- 1061m 48s) (90000 9%) 5.4081
116m 17s (- 1046m 34s) (100000 10%) 5.3800
128m 17s (- 1038m 0s) (110000 11%) 5.3433
140m 12s (- 1028m 14s) (120000 12%) 5.2689
151m 59s (- 1017m 10s) (130000 13%) 5.2705
163m 59s (- 1007m 19s) (140000 14%) 5.2246
175m 58s (- 997m 8s) (150000 15%) 5.2267
187m 50s (- 986m 10s) (160000 16%) 5.1996
199m 44s (- 975m 13s) (170000 17%) 5.1544
211m 15s (- 962m 21s) (180000 18%) 5.1429
222m 13s (- 947m 23s) (190000 19%) 5.1057
233m 14s (- 932m 58s) (200000 20%) 5.1190
244m 8s (- 918m 26s) (210000 21%) 5.0743
255m 11s (- 904m 45s) (220000 22%) 5.0937
266m 16s (- 891m 27s) (230000 23%) 5.0530
277m 23s (- 878m 23s) (240000 24%) 5.0080
289m 25s (- 868m 16s) (250000 25%) 5.0381
301m 28s (- 858m 3s) (260000 26%) 5.0423
313m 30s (- 847m 38s) (270000 27%) 5.0130
325m 38s (- 837m 21s) (280000 28%) 5.0144
337m 36s (- 826m 34s) (290000 28%) 4.9949
348m 36s (- 813m 25s) (300000 30%) 4.9513
359m 47s (- 800m 48s) (310000 31%) 4.9825
370m 50s (- 788m 2s) (320000 32%) 4.9876
381m 54s (- 775m 22s) (330000 33%) 4.9574
392m 56s (- 762m 45s) (340000 34%) 4.9532
404m 0s (- 750m 18s) (350000 35%) 4.9137
415m 52s (- 739m 20s) (360000 36%) 4.9051
427m 58s (- 728m 43s) (370000 37%) 4.9156
440m 8s (- 718m 7s) (380000 38%) 4.9394
452m 14s (- 707m 21s) (390000 39%) 4.9333
464m 25s (- 696m 38s) (400000 40%) 4.8760
475m 40s (- 684m 30s) (410000 41%) 4.8906
486m 51s (- 672m 19s) (420000 42%) 4.8922
498m 0s (- 660m 8s) (430000 43%) 4.8364
509m 13s (- 648m 6s) (440000 44%) 4.8890
520m 20s (- 635m 58s) (450000 45%) 4.9041
532m 4s (- 624m 36s) (460000 46%) 4.9055
544m 14s (- 613m 43s) (470000 47%) 4.8556
556m 29s (- 602m 52s) (480000 48%) 4.8626
568m 44s (- 591m 57s) (490000 49%) 4.8434
580m 49s (- 580m 49s) (500000 50%) 4.8369
592m 23s (- 569m 9s) (510000 51%) 4.8748
603m 25s (- 557m 0s) (520000 52%) 4.8593
614m 33s (- 544m 59s) (530000 53%) 4.8602
```

```
625m 48s (- 533m 5s) (540000 54%) 4.8512
636m 56s (- 521m 8s) (550000 55%) 4.8698
648m 24s (- 509m 27s) (560000 56%) 4.8280
660m 29s (- 498m 16s) (570000 56%) 4.8836
672m 37s (- 487m 4s) (580000 57%) 4.8821
684m 41s (- 475m 48s) (590000 59%) 4.8859
696m 51s (- 464m 34s) (600000 60%) 4.8674
708m 32s (- 453m 0s) (610000 61%) 4.8710
719m 44s (- 441m 8s) (620000 62%) 4.8344
730m 58s (- 429m 18s) (630000 63%) 4.8232
742m 14s (- 417m 30s) (640000 64%) 4.8510
753m 27s (- 405m 42s) (650000 65%) 4.8242
765m 20s (- 394m 16s) (660000 66%) 4.8124
777m 30s (- 382m 56s) (670000 67%) 4.8324
789m 43s (- 371m 38s) (680000 68%) 4.8606
801m 44s (- 360m 12s) (690000 69%) 4.8504
813m 50s (- 348m 47s) (700000 70%) 4.8672
825m 14s (- 337m 4s) (710000 71%) 4.8549
836m 26s (- 325m 17s) (720000 72%) 4.8367
847m 33s (- 313m 28s) (730000 73%) 4.8342
858m 43s (- 301m 42s) (740000 74%) 4.8123
870m 1s (- 290m 0s) (750000 75%) 4.8375
881m 14s (- 278m 17s) (760000 76%) 4.8244
893m 0s (- 266m 44s) (770000 77%) 4.8570
905m 5s (- 255m 16s) (780000 78%) 4.8568
917m 15s (- 243m 49s) (790000 79%) 4.8344
929m 25s (- 232m 21s) (800000 80%) 4.8750
941m 40s (- 220m 53s) (810000 81%) 4.8312
953m 12s (- 209m 14s) (820000 82%) 4.8010
964m 28s (- 197m 32s) (830000 83%) 4.8378
975m 40s (- 185m 50s) (840000 84%) 4.8037
986m 48s (- 174m 8s) (850000 85%) 4.8871
998m 2s (- 162m 28s) (860000 86%) 4.8305
1009m 25s (- 150m 50s) (870000 87%) 4.8382
1021m 40s (- 139m 19s) (880000 88%) 4.8170
1033m 56s (- 127m 47s) (890000 89%) 4.8284
1046m 16s (- 116m 15s) (900000 90%) 4.8255
1058m 32s (- 104m 41s) (910000 91%) 4.8366
1070m 21s (- 93m 4s) (920000 92%) 4.7955
1081m 31s (- 81m 24s) (930000 93%) 4.8202
1092m 42s (- 69m 44s) (940000 94%) 4.8265
1103m 53s (- 58m 5s) (950000 95%) 4.7972
1116m 8s (- 46m 30s) (960000 96%) 4.8123
1128m 21s (- 34m 53s) (970000 97%) 4.8316
1140m 35s (- 23m 16s) (980000 98%) 4.8094
1152m 47s (- 11m 38s) (990000 99%) 4.7914
1164m 46s (- 0m 0s) (1000000 100%) 4.8112
<Figure size 640x480 with 0 Axes>
```

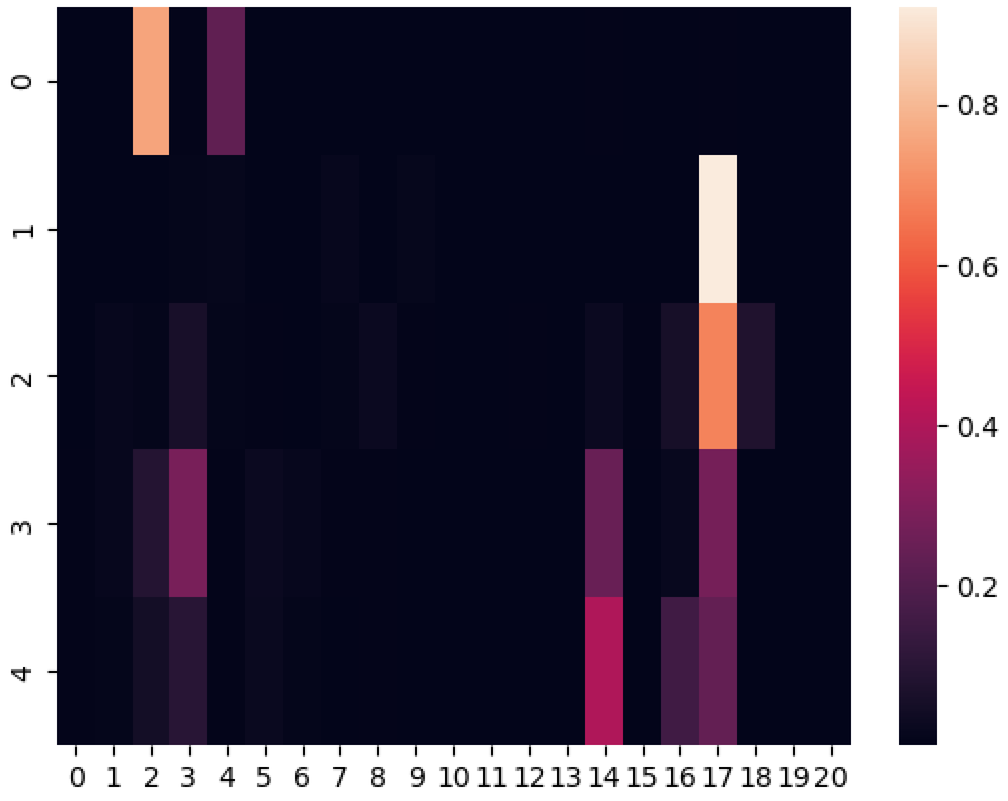


In [20]:
```
evaluateRandomly(encoder1, attn_decoder1)
```

```
> Most of the mughal gardens are rectangularat the whoose center there is a Paviliontomb
= अधिकतर मुगल चारबाग आयताकार होते हैं जिनके केन्द्र में एक मण्डपमकबरा बना होता है।
< मुगल मुगल मुगल मुगल का केन्द्र हैं <EOS>

> What are the professed war aims of the Western Allies
= पश्चिमी मित्रराष्ट्रों के युद्ध के घोषित उद्देश्य क्या हैं
< युद्ध का का युद्ध क्या का है  <EOS>

> because you know that if everything were free
= क्योंकि आप जानते हैं कि अगर सब कुछ मुफ्त होता
< अगर आप जानते हैं कि सब सब जानते हैं <EOS>

> These are hugely powerful forces
= ये सब बहुत ही शक्तिशाली ताकतें हैं
< ये बहुत बहुत बहुत हैं <EOS>

> Now Im not a lawyer
= अब मैं एक वकील नहीं हूँ
< मैं अब एक नहीं नहीं <EOS>

> and to stop controlling and predicting
= और नियंत्रण करना और अनुमान लगाना बंद करना ।
< और और और करने और करने करने लिए <EOS>

> I told you Theyll go there
= मैंने कहा था  ये वहां जायेंगे
< मैंने कहा “मैं आपको पता <EOS>

> in a secret meeting the decided to assassinate police superintendent sadirs
= एक गुप्त योजना के तहत इन्होंने पुलिस सुपरिंटेंडेंट सैंडर्स को मारने की सोची ।
< इस में में सुभाषबाबू को को को की की की की <EOS>

> But are you sure you are going to need all those extra features
= पर क्या आपको ठीक से पता है कि आपको उन सभी अतिरिक्त सुविधाओं की ज़रूरत है
< लेकिन आप सभी कि कि कि की की की की हैं  <EOS>

> Roy never regarded the Congress as the political party of the bourgeoisie
```

```
= राय कांग्रेस को भी अभिजातवर्गीय पार्टी नहीं समझते थे
< राय राय राय राय  पार्टी के पार्टी नहीं  नहीं  <EOS>
```

In [21]:
```python
import seaborn as sns
output_words, attentions = evaluate(encoder1, attn_decoder1, "Hello how are you")
ax = sns.heatmap(attentions.numpy())
plt.show()
```

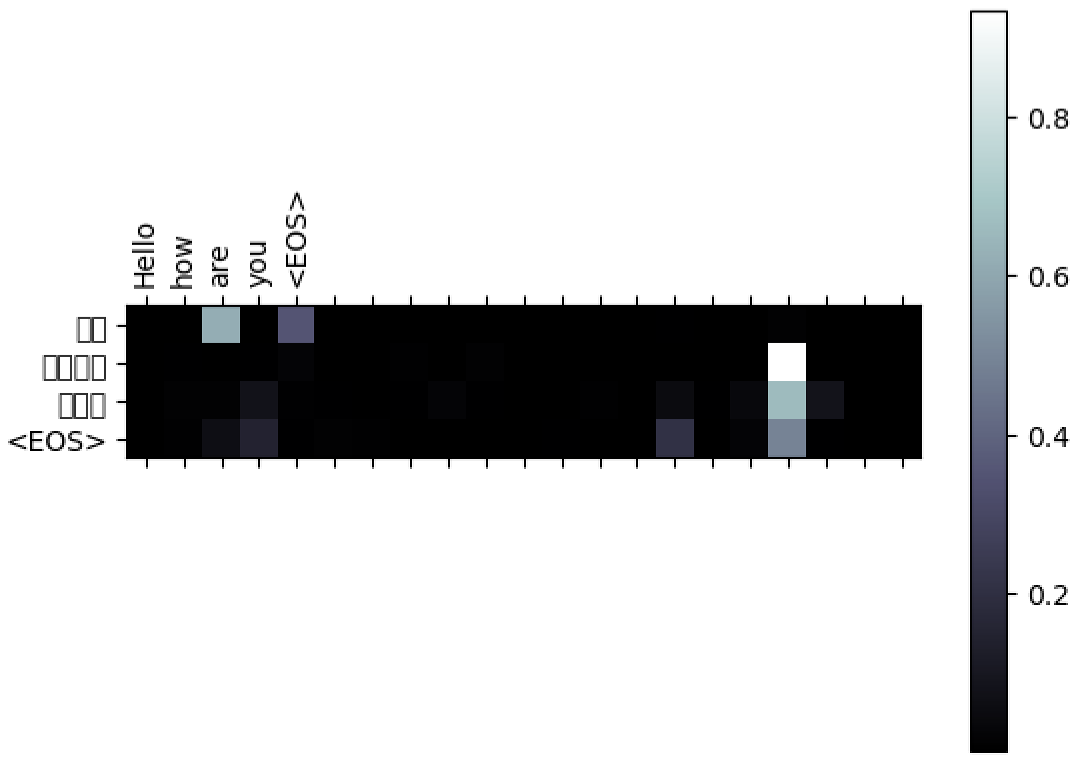


In [22]:
```python
def showAttention(input_sentence, output_words, attentions):
    # Set up figure with colorbar
    fig = plt.figure()
    ax = fig.add_subplot(111)
    cax = ax.matshow(attentions.numpy(), cmap='bone')
    fig.colorbar(cax)

    # Set up axes
    ax.set_xticklabels([''] + input_sentence.split(' ') +
                       ['<EOS>'], rotation=90)
    ax.set_yticklabels([''] + output_words)

    # Show label at every tick
    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

    plt.show()


def evaluateAndShowAttention(input_sentence):
    output_words, attentions = evaluate(
        encoder1, attn_decoder1, input_sentence)
    print('input =', input_sentence)
    print('output =', ' '.join(output_words))
    showAttention(input_sentence, output_words, attentions)
```

In [23]:
```python
evaluateAndShowAttention("Hello how are you")
```

```
input = Hello how are you
output = आप कैसे हैं <EOS>
```

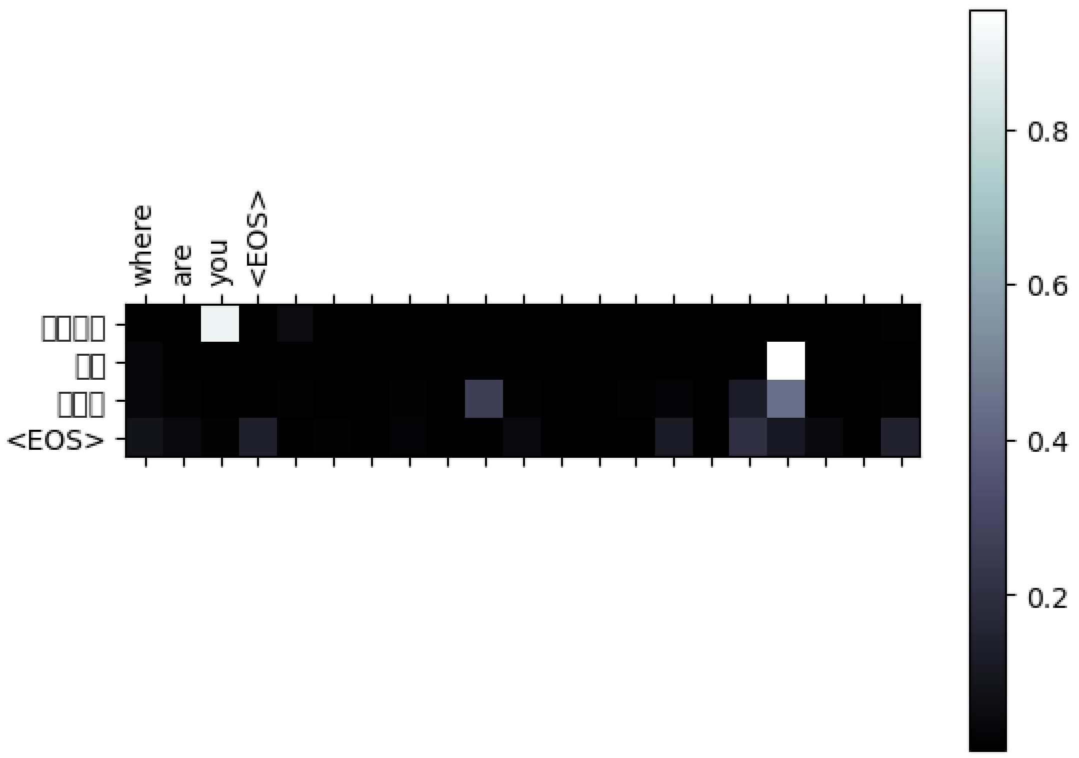
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2310 (\N{DEVANAGARI LETTER AA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Matplotlib currently does not support Devanagari natively.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2346 (\N{DEVANAGARI LETTER PA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2325 (\N{DEVANAGARI LETTER KA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2376 (\N{DEVANAGARI VOWEL SIGN AI}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2360 (\N{DEVANAGARI LETTER SA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2375 (\N{DEVANAGARI VOWEL SIGN E}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2361 (\N{DEVANAGARI LETTER HA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2306 (\N{DEVANAGARI SIGN ANUSVARA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [24]:
```
evaluateAndShowAttention("where are you")
```

```
input = where are you
output = जहाँ आप हैं <EOS>
```

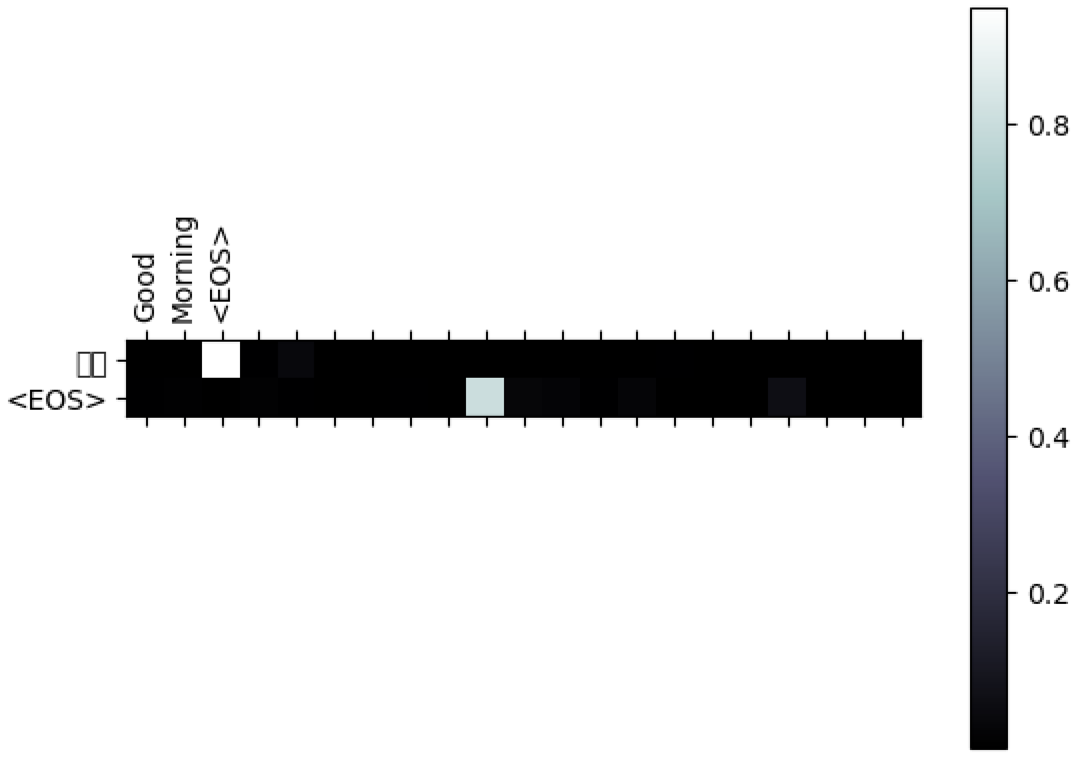
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2332 (\N{DEVANAGARI LETTER JA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2366 (\N{DEVANAGARI VOWEL SIGN AA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2305 (\N{DEVANAGARI SIGN CANDRABINDU}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [25]: evaluateAndShowAttention("Good Morning")
```

```
input = Good Morning
output = इस <EOS>
```

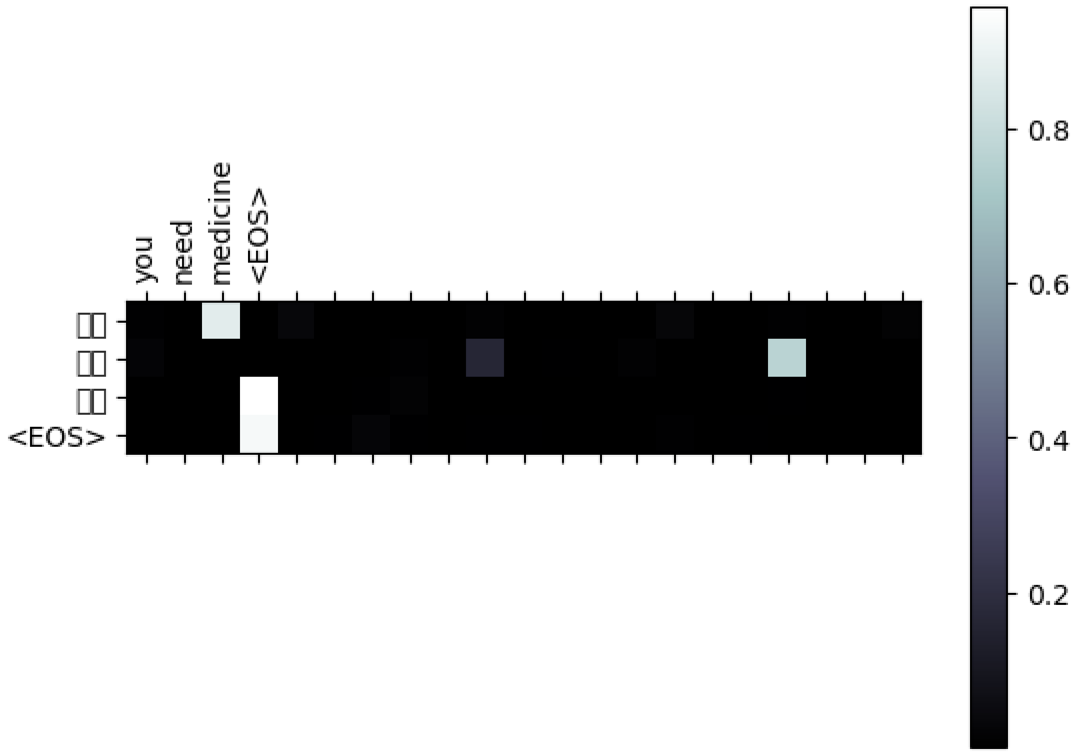
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2311 (\N{DEVANAGARI LETTER I}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [26]: evaluateAndShowAttention("you need medicine")
```

```
input = you need medicine
output = आप के के <EOS>
```

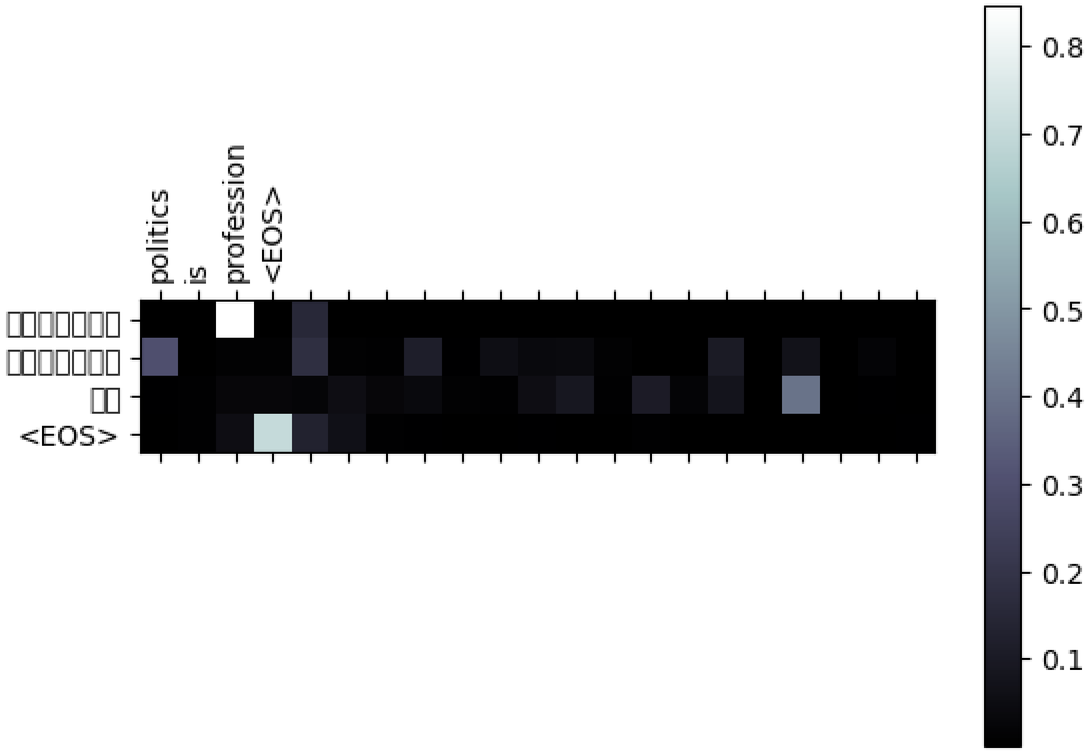
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
```



```
In [27]: evaluateAndShowAttention("politics is profession")
```

```
input = politics is profession
output = राजनीति राजनीति है <EOS>
```

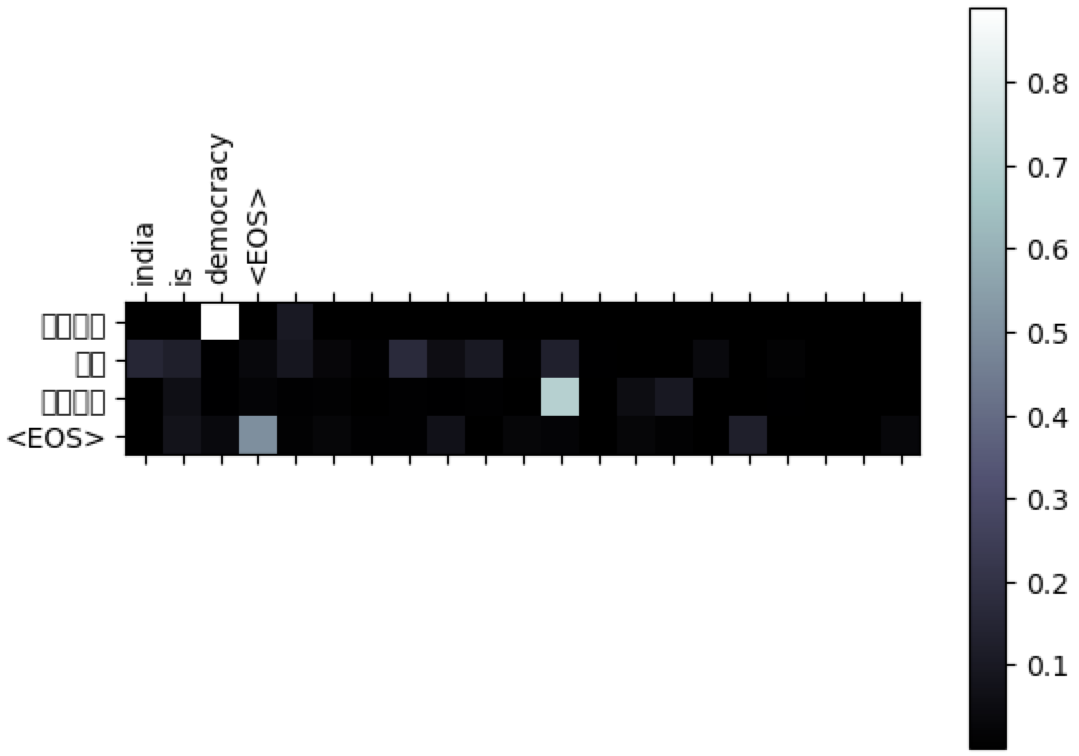
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2352 (\N{DEVANAGARI LETTER RA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2344 (\N{DEVANAGARI LETTER NA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2368 (\N{DEVANAGARI VOWEL SIGN II}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2340 (\N{DEVANAGARI LETTER TA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2367 (\N{DEVANAGARI VOWEL SIGN I}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



In [28]:
```
evaluateAndShowAttention("india is democracy")
```

```
input = india is democracy
output = भारत का भारत <EOS>
```

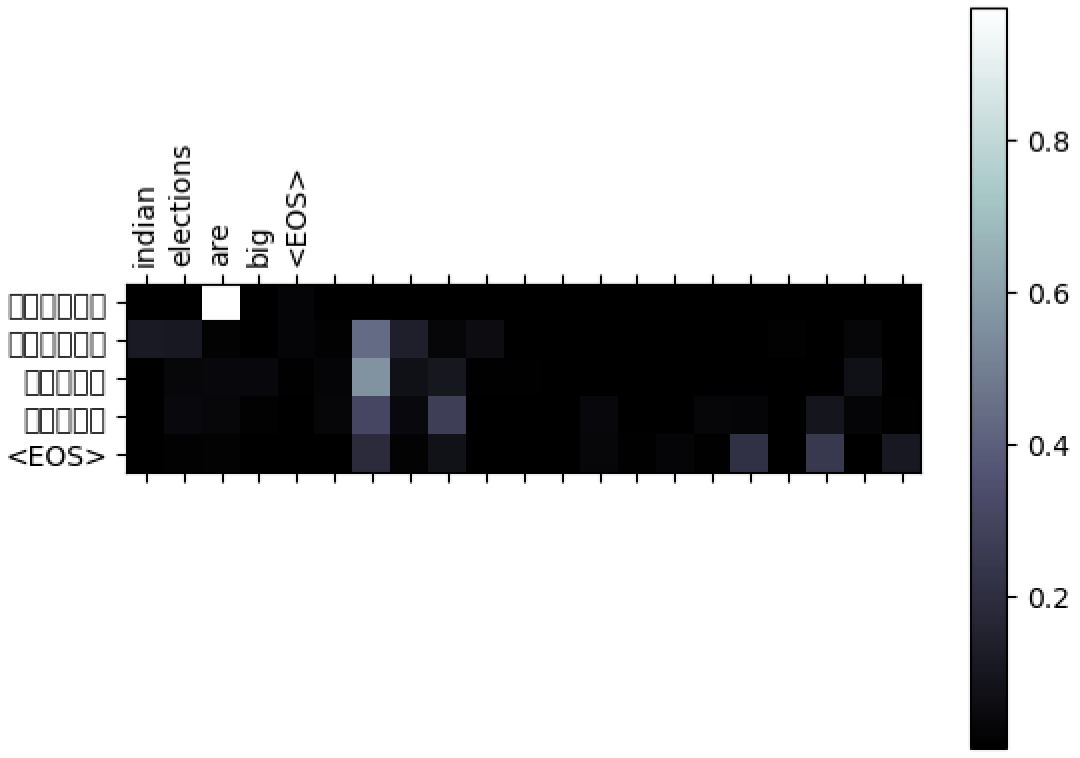
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2349 (\N{DEVANAGARI LETTER BHA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [29]: evaluateAndShowAttention("indian elections are big")
```

```
input = indian elections are big
output = भारतीय भारतीय चुनाव चुनाव <EOS>
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:9: UserWarning: FixedForm
atter should only be used together with FixedLocator
  ax.set_xticklabels([''] + input_sentence.split(' ') +
C:\Users\Admin\AppData\Local\Temp\ipykernel_8720\3159908345.py:11: UserWarning: FixedFor
matter should only be used together with FixedLocator
  ax.set_yticklabels([''] + output_words)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2351 (\N{DEVANAGARI LETTER YA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2330 (\N{DEVANAGARI LETTER CA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2369 (\N{DEVANAGARI VOWEL SIGN U}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
C:\Users\Admin\.conda\envs\torch\lib\site-packages\IPython\core\pylabtools.py:152: UserW
arning: Glyph 2357 (\N{DEVANAGARI LETTER VA}) missing from current font.
  fig.canvas.print_figure(bytes_io, **kw)
```



```
In [ ]:
```