overview_EDA_pipeline_model_Final

February 4, 2021

Problem Overview: Spelling error correction is the task of automatically correcting spelling errors in text; e.g. [I followed his advcie -> I followed his advice]. It can be used to not only help language learners improve their writing erros, but also alert native speakers to accidental mistakes or typos. The aim of the task is to produce a correctly spelled sentence given a incorrectly spelled sentence.

Business Problem: By building the automated spelling error correction. We can create automated tools for writing English scientific texts, Filtering out sentences that need spelling improvements, evaluating articles, etc. These all mentioned tasks are done manually so, by automating this process we can save both time and money for the company.

ML formulation: Building a model using techniques like Encoder-Decoder architecture, Bidirectional LSTM with attention mechanism, etc. can be used.

Performance metric: BLEU Score Bilingual Evaluation Understudy(BLEU) is a metric for comparing machine translated sentence to one or more reference setences. The metric ranges on scale of 0 to 1, in an attempt to measure the adequacy and fluency of the MT output. the more overlap there is with their human reference translations and thus, the better the translation. The BLEU is programming task to compare n-grams of the translated sentences with the reference sentence and count the number of matches. The more the matches the better the performance.

[1]:
```
import os
import re
import numpy as np
import pandas as pd
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf

from tqdm import tqdm
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import*
from nltk.translate.bleu_score import sentence_bleu
```

```
warnings.filterwarnings('ignore')
```

1 EDA

Preprocessing data: Text files which we need to combine to introduce spelling errors in the dataset. So, that we can train model for spelling error correction.

[2]:
```
text_files = os.listdir('data')
text_files
```

[2]:
```
['nietzsche.txt',
 'pride_and_prejudice.txt',
 'shakespeare.txt',
 'war_and_peace.txt',
 'wonderland.txt']
```

[3]:
```
REMOVE_CHARS = '[#$%"\+@<=>!&,-.?:;()*\[\]\'^_`{|}~/\d\t\r\x0b\x0c]'
train_text = ''
for filename in text_files[:-1]:
    file = open('data/'+filename).read().split()
    train_text += ' '.join([re.sub(REMOVE_CHARS, '', token) for token in␣
 ↪file])+' '
```

[4]:
```
from nltk import ngrams

unigram_train_data = set([token for token in train_text.split()])
unigram_train_data = list(filter(None, set(unigram_train_data)))

bigram_train_data = list(ngrams(train_text.split(), 2))
bigram_train_data = [' '.join(x) for x in bigram_train_data]

trigram_train_data = list(ngrams(train_text.split(), 3))
trigram_train_data = [' '.join(x) for x in trigram_train_data]

print('Size of unigram train data:', len(unigram_train_data))
print('Size of bigram train data:', len(bigram_train_data))
print('Size of trigram train data:', len(trigram_train_data))
```

```
Size of unigram train data: 36821
Size of bigram train data: 983848
Size of trigram train data: 983847
```

[5]:
```
file = open('data/'+text_files[-1]).read().split()
test_text = ' '.join([re.sub(REMOVE_CHARS, '', token) for token in file])
```

[6]:
```
from nltk import ngrams

unigram_test_data = set([token for token in test_text.split()])
```

```
unigram_test_data = list(filter(None, set(unigram_test_data)))

bigram_test_data = list(ngrams(test_text.split(), 2))
bigram_test_data = [' '.join(x) for x in bigram_test_data]

trigram_test_data = list(ngrams(test_text.split(), 3))
trigram_test_data = [' '.join(x) for x in trigram_test_data]

print('Size of unigram test data:', len(unigram_test_data))
print('Size of bigram test data:', len(bigram_test_data))
print('Size of trigram test data:', len(trigram_test_data))
```

```
Size of unigram test data: 3152
Size of bigram test data: 26387
Size of trigram test data: 26386
```

[7]:
```
unigram_length = []
for i in unigram_train_data:
    unigram_length.append(len(i))

bigram_length = []
for i in bigram_train_data:
    bigram_length.append(len(i))

trigram_length = []
for i in trigram_train_data:
    trigram_length.append(len(i))
```

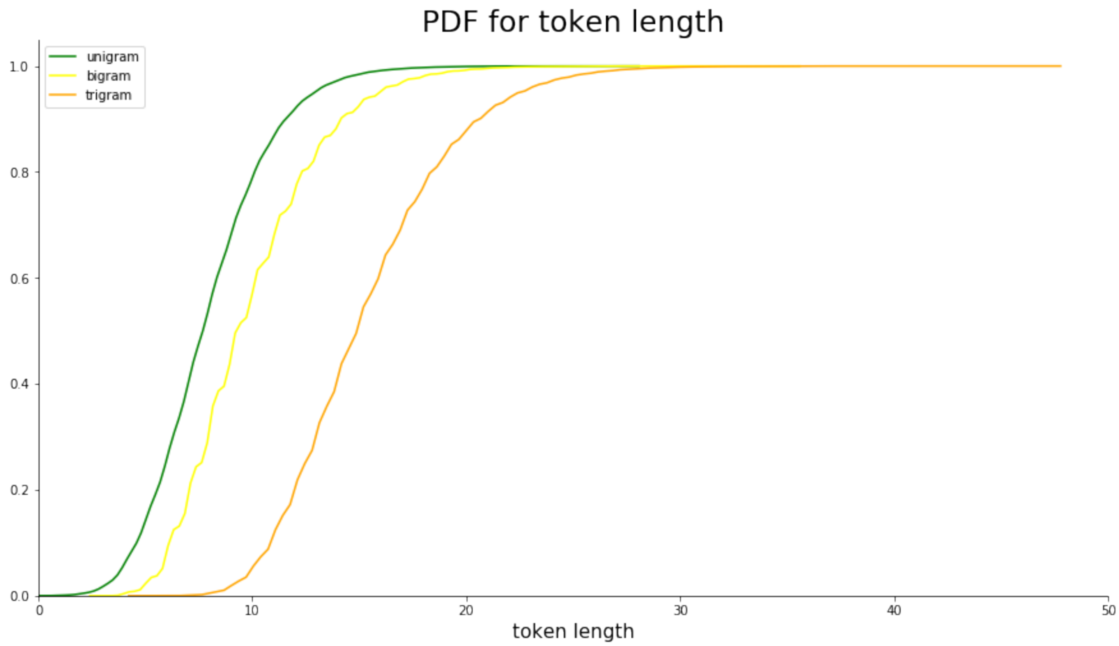
[8]:
```
for i in range(0,101,10):
    print(i,np.percentile(unigram_length, i), np.percentile(bigram_length, i),␣
 ↪np.percentile(trigram_length, i))
for i in range(90,101):
    print(i,np.percentile(unigram_length, i), np.percentile(bigram_length, i),␣
 ↪np.percentile(trigram_length, i))
for i in [99.1,99.2,99.3,99.4,99.5,99.6,99.7,99.8,99.9,100]:
    print(i,np.percentile(unigram_length, i), np.percentile(bigram_length, i),␣
 ↪np.percentile(trigram_length, i))
```

```
0 1.0 3.0 5.0
10 5.0 6.0 11.0
20 6.0 7.0 12.0
30 6.0 8.0 13.0
40 7.0 9.0 14.0
50 8.0 9.0 15.0
60 8.0 10.0 16.0
70 9.0 11.0 17.0
80 10.0 12.0 18.0
90 12.0 14.0 21.0
```

```
100 27.0 35.0 47.0
90 12.0 14.0 21.0
91 12.0 14.0 21.0
92 12.0 15.0 21.0
93 12.0 15.0 22.0
94 12.0 15.0 22.0
95 13.0 16.0 22.0
96 13.0 16.0 23.0
97 14.0 17.0 24.0
98 14.0 18.0 25.0
99 16.0 19.0 26.0
100 27.0 35.0 47.0
99.1 16.0 20.0 27.0
99.2 16.0 20.0 27.0
99.3 16.0 20.0 27.0
99.4 17.0 20.0 28.0
99.5 17.0 21.0 28.0
99.6 17.0 21.0 28.0
99.7 18.0 22.0 29.0
99.8 19.0 22.0 30.0
99.9 20.0 24.0 32.0
100 27.0 35.0 47.0
```

[9]:

```
plt.figure(figsize = (15,8))
ax = sns.kdeplot(data = unigram_length, color='green', cumulative = True, label␣
↪= 'unigram')
sns.kdeplot(data = bigram_length, color='yellow', cumulative = True, ax = ax,␣
↪label = 'bigram')
sns.kdeplot(data = trigram_length, color='orange', cumulative = True, ax = ax,␣
↪label = 'trigram')
plt.xlabel('token length', fontdict = {'fontsize':15})
plt.title('PDF for token length', fontdict = {'fontsize': 23})
ax.legend()
plt.xlim(0, 50)
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
```



Aim : Plot for determining the distribution token length of unigram, bigram, and trigram dataset.

Conclusion : 1. 99.9% of unigram token have length less than 20 characters. 2. 99.9% of biigram token have length less than 24 characters. 3. 99.9% of trigram token have length less than 32 characters.

[10]:
```
def add_speling_errors(token, error_rate, VOCAB):
    """Add some artificial spelling mistakes."""
    assert(0.0 <= error_rate < 1.0)
    if len(token) < 3:
        return token
    rand = np.random.rand()
    # Here are 4 different ways spelling mistakes can occur,
    # each of which has equal chance.
    prob = error_rate / 4.0
    if rand < prob:
        # Replace a character with a random character.
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + np.random.choice(VOCAB) \
                + token[random_char_index + 1:]
    elif prob < rand < prob * 2:
        # Delete a character.
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + token[random_char_index + 1:]
    elif prob * 2 < rand < prob * 3:
        # Add a random character.
```

```
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + np.random.choice(VOCAB) \
                + token[random_char_index:]
    elif prob * 3 < rand < prob * 4:
        # Transpose 2 characters.
        random_char_index = np.random.randint(len(token) - 1)
        token = token[:random_char_index]  + token[random_char_index + 1] \
                + token[random_char_index] + token[random_char_index + 2:]
    else:
        # No spelling errors.
        pass
    return token
```

[2]:
```
UNIGRAM_VOCAB = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',␣
↪'m', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', \
                 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',␣
↪'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', \
                 '<SOW>', '<EOW>']

NGRAM_VOCAB = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',␣
↪'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', \
               'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',␣
↪'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', \
               ' ', '<SOW>', '<EOW>']
```

[12]:
```
unigram_maxlen = 20
unigram_train = []
for token in unigram_train_data:
    if len(token) < unigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, UNIGRAM_VOCAB)]
        unigram_train.append(point)
unigram_train = pd.DataFrame(unigram_train, columns = ['output', 'input'])
unigram_train.head()
```

[12]:

	output	input
0	magistrates	magistratse
1	Cookshops	Cookshops
2	aristocratic	arstocratic
3	graced	graced
4	wagging	wagigng

[13]:
```
bigram_maxlen = 24
bigram_train = []
for token in bigram_train_data:
    if len(token) < bigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, NGRAM_VOCAB)]
        bigram_train.append(point)
```

```
bigram_train = pd.DataFrame(bigram_train, columns = ['output', 'input'])
bigram_train.head()
```

[13]:

	output	input
0	PREFACE SUPPOSING	PREFACE SUPPOSING
1	SUPPOSING that	SUPPOSINGt hat
2	that Truth	that TruEh
3	Truth is	Truth is
4	is a	si a

[14]:
```
trigram_maxlen = 32
trigram_train = []
for token in trigram_train_data:
    if len(token) < trigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, NGRAM_VOCAB)]
        trigram_train.append(point)
trigram_train = pd.DataFrame(trigram_train, columns = ['output', 'input'])
trigram_train.head()
```

[14]:

	output	input
0	PREFACE SUPPOSING that	PREFACE SUPPOSIN that
1	SUPPOSING that Truth	SUPPOSsING that Truth
2	that Truth is	that Truth is
3	Truth is a	Truth si a
4	is a womanwhat	is a womanwhat

[15]:
```
unigram_train['enc_inp'] = unigram_train['input'].astype(str).apply(lambda x:␣
↪'<SOW> '+' '.join(list(x))+' <EOW>')
unigram_train['dec_inp'] = unigram_train['output'].astype(str).apply(lambda x:␣
↪'<SOW> '+' '.join(list(x)))
unigram_train['dec_out'] = unigram_train['output'].astype(str).apply(lambda x:␣
↪' '.join(list(x))+' <EOW>')

bigram_train['enc_inp'] = bigram_train['input'].astype(str).apply(lambda x:␣
↪'<SOW>*'+'*'.join(list(x))+'*<EOW>')
bigram_train['dec_inp'] = bigram_train['output'].astype(str).apply(lambda x:␣
↪'<SOW>*'+'*'.join(list(x)))
bigram_train['dec_out'] = bigram_train['output'].astype(str).apply(lambda x:␣
↪'*'.join(list(x))+'*<EOW>')

trigram_train['enc_inp'] = trigram_train['input'].astype(str).apply(lambda x:␣
↪'<SOW>*'+'*'.join(list(x))+'*<EOW>')
trigram_train['dec_inp'] = trigram_train['output'].astype(str).apply(lambda x:␣
↪'<SOW>*'+'*'.join(list(x)))
trigram_train['dec_out'] = trigram_train['output'].astype(str).apply(lambda x:␣
↪'*'.join(list(x))+'*<EOW>')
```

[16]:
```
unigram_train, unigram_val = train_test_split(unigram_train, test_size = 0.1,␣
 ↪random_state = 0)
bigram_train, bigram_val = train_test_split(bigram_train, test_size = 0.1,␣
 ↪random_state = 0)
trigram_train, trigram_val = train_test_split(trigram_train, test_size = 0.1,␣
 ↪random_state = 0)
```

[17]:
```
unigram_test = []
for token in unigram_test_data:
    if len(token) < unigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, UNIGRAM_VOCAB)]
        unigram_test.append(point)
unigram_test = pd.DataFrame(unigram_test, columns = ['output', 'input'])
unigram_test.head()
```

[17]:
```
       output       input
0    slatesll    slatesll
1    rumbling    rmubling
2      chance      chacne
3      bright     brnight
4  ridiculous  ridiculous
```

[18]:
```
bigram_test = []
for token in bigram_test_data:
    if len(token) < bigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, NGRAM_VOCAB)]
        bigram_test.append(point)
bigram_test = pd.DataFrame(bigram_test, columns = ['output', 'input'])
bigram_test.head()
```

[18]:
```
                 output                  input
0  ï»¿ALICES ADVENTURES  ï»¿ALIECS ADVENTURES
1         ADVENTURES IN          ADVENTURES IN
2         IN WONDERLAND          IN WONDEtLAND
3      WONDERLAND Lewis       WONDERLAND Lewis
4          Lewis Carroll          Lewis Carrlol
```

[19]:
```
trigram_test = []
for token in trigram_test_data:
    if len(token) < unigram_maxlen:
        point = [token, add_speling_errors(token, 0.5, NGRAM_VOCAB)]
        trigram_test.append(point)
trigram_test = pd.DataFrame(trigram_test, columns = ['output', 'input'])
trigram_test.head()
```

[19]:
```
                output                  input
0  IN WONDERLAND Lewis  IN WONDERoLAND Lewis
```

```
1    Lewis Carroll THE     Lewis Carroll THE
2    EDITION CHAPTER I     EDITION CHATPER I
3       CHAPTER I Down        CHAPTRE I Down
4           I Down the            I Down the
```

[100]:
```python
unigram_train.to_csv('unigram_train.csv')
unigram_test.to_csv('unigram_test.csv')
unigram_val.to_csv('unigram_val.csv')

bigram_train.to_csv('bigram_train.csv')
bigram_test.to_csv('bigram_test.csv')
bigram_val.to_csv('bigram_val.csv')

trigram_train.to_csv('trigram_train.csv')
trigram_test.to_csv('trigram_test.csv')
trigram_val.to_csv('trigram_val.csv')
```

[20]:
```python
print('Shape of unigram train set :',unigram_train.shape)
print('Shape of unigram test set :',unigram_test.shape)
print('Shape of unigram validation set :', unigram_val.shape)

print('\nShape of bigram train set :',bigram_train.shape)
print('Shape of bigram test set :',bigram_test.shape)
print('Shape of bigram validation set :', bigram_val.shape)

print('\nShape of trigram train set :',trigram_train.shape)
print('Shape of trigram test set :',trigram_test.shape)
print('Shape of trigram validation set :', trigram_val.shape)
```

```
Shape of unigram train set : (33101, 5)
Shape of unigram test set : (3150, 2)
Shape of unigram validation set : (3678, 5)

Shape of bigram train set : (884533, 5)
Shape of bigram test set : (26377, 2)
Shape of bigram validation set : (98282, 5)

Shape of trigram train set : (884550, 5)
Shape of trigram test set : (24586, 2)
Shape of trigram validation set : (98284, 5)
```

[21]:
```python
print(unigram_train.isnull().sum(), unigram_val.isnull().sum(), unigram_test.
 ↪isnull().sum())
print(bigram_train.isnull().sum(), bigram_val.isnull().sum(), bigram_test.
 ↪isnull().sum())
print(trigram_train.isnull().sum(), trigram_val.isnull().sum(), trigram_test.
 ↪isnull().sum())
```

```
output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output    0
input     0
dtype: int64
output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output    0
input     0
dtype: int64
output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output     0
input      0
enc_inp    0
dec_inp    0
dec_out    0
dtype: int64 output    0
input     0
dtype: int64
```

2 Modeling

[14]:

```
unigram_train = pd.read_csv('unigram_train.csv', index_col = 0)
unigram_test = pd.read_csv('unigram_test.csv', index_col = 0)
unigram_val = pd.read_csv('unigram_val.csv', index_col = 0)

bigram_train = pd.read_csv('bigram_train.csv', index_col = 0)
```

```
bigram_test = pd.read_csv('bigram_test.csv', index_col = 0)
bigram_val = pd.read_csv('bigram_val.csv', index_col = 0)

trigram_train = pd.read_csv('trigram_train.csv', index_col = 0)
trigram_test = pd.read_csv('trigram_test.csv', index_col = 0)
trigram_val = pd.read_csv('trigram_val.csv', index_col = 0)
```

[16]:
```
def split_on_star(input_data):
    return tf.strings.split(input_data, sep = '*')
```

[15]:
```
batch_size = 128
```

[11]:
```
unigram_maxlen = 20
bigram_maxlen = 24
trigram_maxlen = 32
```

[17]:
```
unigram_vec = TextVectorization(output_sequence_length= unigram_maxlen+2,␣
 ↪standardize = None, split='whitespace', max_tokens = len(UNIGRAM_VOCAB)+2,␣
 ↪output_mode='int')
unigram_vec.adapt(UNIGRAM_VOCAB)

bigram_vec = TextVectorization(output_sequence_length= bigram_maxlen+2,␣
 ↪standardize = None, split = split_on_star, max_tokens = len(NGRAM_VOCAB)+2,␣
 ↪output_mode='int')
bigram_vec.adapt(NGRAM_VOCAB)

trigram_vec = TextVectorization(output_sequence_length= trigram_maxlen+2,␣
 ↪standardize = None, split = split_on_star, max_tokens = len(NGRAM_VOCAB)+2,␣
 ↪output_mode='int')
trigram_vec.adapt(NGRAM_VOCAB)

unigram_index_to_word = {idx: word for idx, word in enumerate(unigram_vec.
 ↪get_vocabulary())}
unigram_word_to_index = {word: idx for idx, word in enumerate(unigram_vec.
 ↪get_vocabulary())}

bigram_index_to_word = {idx: word for idx, word in enumerate(bigram_vec.
 ↪get_vocabulary())}
bigram_word_to_index = {word: idx for idx, word in enumerate(bigram_vec.
 ↪get_vocabulary())}

trigram_index_to_word = {idx: word for idx, word in enumerate(trigram_vec.
 ↪get_vocabulary())}
trigram_word_to_index = {word: idx for idx, word in enumerate(trigram_vec.
 ↪get_vocabulary())}
```

```python
def unigram_mapping(x):
    enc_inp = unigram_vec(x[:, 2])
    dec_inp = unigram_vec(x[:, 3])
    dec_out = unigram_vec(x[:, 4])
    return (enc_inp, dec_inp), dec_out

def bigram_mapping(x):
    enc_inp = bigram_vec(x[:, 2])
    dec_inp = bigram_vec(x[:, 3])
    dec_out = bigram_vec(x[:, 4])
    return (enc_inp, dec_inp), dec_out

def trigram_mapping(x):
    enc_inp = trigram_vec(x[:, 2])
    dec_inp = trigram_vec(x[:, 3])
    dec_out = trigram_vec(x[:, 4])
    return (enc_inp, dec_inp), dec_out

unigram_train_dataset = tf.data.Dataset.from_tensor_slices(unigram_train.
 ↪values).repeat().batch(batch_size).map(unigram_mapping).prefetch(1)
unigram_val_dataset = tf.data.Dataset.from_tensor_slices(unigram_val.values).
 ↪repeat().batch(batch_size).map(unigram_mapping).prefetch(1)

bigram_train_dataset = tf.data.Dataset.from_tensor_slices(bigram_train.values).
 ↪repeat().batch(batch_size).map(bigram_mapping).prefetch(1)
bigram_val_dataset = tf.data.Dataset.from_tensor_slices(bigram_val.values).
 ↪repeat().batch(batch_size).map(bigram_mapping).prefetch(1)

trigram_train_dataset = tf.data.Dataset.from_tensor_slices(trigram_train.
 ↪values).repeat().batch(batch_size).map(trigram_mapping).prefetch(1)
trigram_val_dataset = tf.data.Dataset.from_tensor_slices(trigram_val.values).
 ↪repeat().batch(batch_size).map(trigram_mapping).prefetch(1)

a = unigram_train_dataset.as_numpy_iterator()
next(a)
```

```
[17]: ((array([[54, 26, 23, …,  0,  0,  0],
               [54, 42, 51, …,  0,  0,  0],
               [54, 35,  7, …,  0,  0,  0],
               …,
               [54, 10, 27, …,  0,  0,  0],
               [54, 15,  7, …,  0,  0,  0],
               [54,  7, 14, …,  0,  0,  0]], dtype=int64),
        array([[54, 26, 23, …,  0,  0,  0],
               [54, 42, 33, …,  0,  0,  0],
               [54, 35,  7, …,  0,  0,  0],
               …,
```

```
        [54, 10, 27, …,  0,  0,  0],
        [54, 15,  7, …,  0,  0,  0],
        [54,  7, 14, …,  0,  0,  0]], dtype=int64)),
 array([[26, 23, 22, …,  0,  0,  0],
        [42, 33, 51, …,  0,  0,  0],
        [35,  7, 25, …,  0,  0,  0],
        …,
        [10, 27,  6, …,  0,  0,  0],
        [15,  7, 14, …,  0,  0,  0],
        [ 7, 14, 24, …,  0,  0,  0]], dtype=int64))
```

1. Seq2Seq

[18]:
```python
class Encoder(tf.keras.Model):
    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        self.lstm_size = lstm_size
        #Initialize Embedding layer
        self.enc_embed = Embedding(input_dim = inp_vocab_size, output_dim =␣
 ↪embedding_size, input_length= input_length)
        #Intialize Encoder LSTM layer
        self.enc_lstm = LSTM(lstm_size, return_sequences = True, return_state =␣
 ↪True, dropout = 0.4)

    def call(self,input_sequence,states):
        embedding = self.enc_embed(input_sequence)
        output_state, enc_h, enc_c = self.enc_lstm(embedding, initial_state =␣
 ↪states)
        return output_state, enc_h, enc_c

    def initialize_states(self,batch_size):
        return [tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size,␣
 ↪self.lstm_size))]

class Decoder(tf.keras.Model):
    def __init__(self,out_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        #Initialize Embedding layer
        self.dec_embed = Embedding(input_dim = out_vocab_size, output_dim =␣
 ↪embedding_size, input_length = input_length)
        #Intialize Decoder LSTM layer
        self.dec_lstm = LSTM(lstm_size, return_sequences = True, return_state =␣
 ↪True, dropout = 0.4)

    def call(self,input_sequence, initial_states):
        embedding = self.dec_embed(input_sequence)
```

```python
        output_state, dec_h, dec_c = self.dec_lstm(embedding, initial_state = initial_states)
        return output_state, dec_h, dec_c

class Encoder_decoder(tf.keras.Model):
    def __init__(self,*params):
        super().__init__()
        #Create encoder object
        self.encoder = Encoder(inp_vocab_size = params[0], embedding_size = params[2], lstm_size = params[3], input_length = params[4])
        #Create decoder object
        self.decoder = Decoder(out_vocab_size = params[1], embedding_size = params[2], lstm_size = params[3], input_length = params[5])
        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        self.dense = Dense(params[1], activation='softmax')

    def call(self, params, training = True):
        enc_inp, dec_inp = params[0], params[1]
        # print(enc_inp, dec_inp)
        initial_state = self.encoder.initialize_states(batch_size)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        output, _, _ = self.decoder(dec_inp ,[enc_h, enc_c])
        output = Dropout(0.5)(output)
        return self.dense(output)

class pred_Encoder_decoder(tf.keras.Model):
    def __init__(self,*params):
        super().__init__()
        #Create encoder object
        self.encoder = Encoder(inp_vocab_size = params[0], embedding_size = params[2], lstm_size = params[3], input_length = params[4])
        #Create decoder object
        self.decoder = Decoder(out_vocab_size = params[1], embedding_size = params[2], lstm_size = params[3], input_length = params[5])
        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        self.dense = Dense(params[1], activation='softmax')
        self.word_to_index = params[6]

    def call(self, params):
        enc_inp = params[0]
        initial_state = self.encoder.initialize_states(1)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        pred = tf.expand_dims([self.word_to_index['<SOW>']], 0)
        dec_h = enc_h
        dec_c = enc_c
        all_pred = []
        for t in range(max_len):
```

```
            pred, dec_h,dec_c = self.decoder(pred, [dec_h, dec_c])
            pred = self.dense(pred)
            pred = tf.argmax(pred, axis = -1)
            all_pred.append(pred)
        return all_pred
```

[19]:
```
def predict(seq, vectorizer, index_to_word, gram ='uni'):
    if gram == 'uni':
        seq = ' '.join(list(seq))
        seq = '<SOW> '+seq+' <EOW>'
    else:
        seq = '*'.join(list(seq))
        seq = '<SOW>*'+seq+'*<EOW>'
    seq = vectorizer([seq])
    pred = pred_model.predict(tf.expand_dims(seq, 0))
    output = []
    for i in pred:
        word = index_to_word[i[0][0]]
        if word == '<EOW>':
            break
        output.append(word)
    return ''.join(output)
```

1.1 UniGram

[67]:
```
vocab_size = len(vec.get_vocabulary())
embedding_dim = 100
lstm_size = 256
max_len = 22
```

[20]:
```
model = Encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪max_len, max_len)
model.compile(optimizer = 'Adam',loss = 'sparse_categorical_crossentropy')

callbacks = [ModelCheckpoint('seq2seq.h5', save_best_only= True, verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = unigram_train_dataset,
          steps_per_epoch = unigram_train.shape[0]//batch_size,
          validation_data = unigram_val_dataset,
          validation_steps = unigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
258/258 [==============================] - ETA: 0s - loss: 1.3088
```

```
Epoch 00001: val_loss improved from inf to 1.08792, saving model to seq2seq.h5
258/258 [==============================] - 4s 14ms/step - loss: 1.3088 -
val_loss: 1.0879
Epoch 2/50
258/258 [==============================] - ETA: 0s - loss: 1.0246
Epoch 00002: val_loss improved from 1.08792 to 0.93814, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 1.0246 -
val_loss: 0.9381
Epoch 3/50
256/258 [============================>.] - ETA: 0s - loss: 0.9280
Epoch 00003: val_loss improved from 0.93814 to 0.84624, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.9278 -
val_loss: 0.8462
Epoch 4/50
256/258 [============================>.] - ETA: 0s - loss: 0.8515
Epoch 00004: val_loss improved from 0.84624 to 0.78058, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.8513 -
val_loss: 0.7806
Epoch 5/50
256/258 [============================>.] - ETA: 0s - loss: 0.8017
Epoch 00005: val_loss improved from 0.78058 to 0.72511, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.8014 -
val_loss: 0.7251
Epoch 6/50
256/258 [============================>.] - ETA: 0s - loss: 0.7480
Epoch 00006: val_loss improved from 0.72511 to 0.66346, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.7478 -
val_loss: 0.6635
Epoch 7/50
258/258 [==============================] - ETA: 0s - loss: 0.6937
Epoch 00007: val_loss improved from 0.66346 to 0.60634, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.6937 -
val_loss: 0.6063
Epoch 8/50
256/258 [============================>.] - ETA: 0s - loss: 0.6483
Epoch 00008: val_loss improved from 0.60634 to 0.55700, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.6483 -
val_loss: 0.5570
Epoch 9/50
256/258 [============================>.] - ETA: 0s - loss: 0.5939
Epoch 00009: val_loss improved from 0.55700 to 0.49638, saving model to
```

```
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.5939 -
val_loss: 0.4964
Epoch 10/50
256/258 [============================>.] - ETA: 0s - loss: 0.5503
Epoch 00010: val_loss improved from 0.49638 to 0.45429, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.5504 -
val_loss: 0.4543
Epoch 11/50
256/258 [============================>.] - ETA: 0s - loss: 0.5132
Epoch 00011: val_loss improved from 0.45429 to 0.41674, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.5132 -
val_loss: 0.4167
Epoch 12/50
256/258 [============================>.] - ETA: 0s - loss: 0.4749
Epoch 00012: val_loss improved from 0.41674 to 0.38192, saving model to
seq2seq.h5
258/258 [==============================] - 3s 12ms/step - loss: 0.4748 -
val_loss: 0.3819
Epoch 13/50
256/258 [============================>.] - ETA: 0s - loss: 0.4376
Epoch 00013: val_loss improved from 0.38192 to 0.34618, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.4374 -
val_loss: 0.3462
Epoch 14/50
257/258 [============================>.] - ETA: 0s - loss: 0.3941
Epoch 00014: val_loss improved from 0.34618 to 0.29948, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.3941 -
val_loss: 0.2995
Epoch 15/50
254/258 [============================>.] - ETA: 0s - loss: 0.3567
Epoch 00015: val_loss improved from 0.29948 to 0.27217, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.3562 -
val_loss: 0.2722
Epoch 16/50
255/258 [============================>.] - ETA: 0s - loss: 0.3254
Epoch 00016: val_loss improved from 0.27217 to 0.24162, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.3251 -
val_loss: 0.2416
Epoch 17/50
257/258 [============================>.] - ETA: 0s - loss: 0.2975
Epoch 00017: val_loss improved from 0.24162 to 0.22571, saving model to
```

```
seq2seq.h5
258/258 [==============================] - 4s 14ms/step - loss: 0.2975 -
val_loss: 0.2257
Epoch 18/50
257/258 [============================>.] - ETA: 0s - loss: 0.2744
Epoch 00018: val_loss improved from 0.22571 to 0.20805, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2743 -
val_loss: 0.2080
Epoch 19/50
257/258 [============================>.] - ETA: 0s - loss: 0.2558
Epoch 00019: val_loss improved from 0.20805 to 0.19446, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2557 -
val_loss: 0.1945
Epoch 20/50
256/258 [============================>.] - ETA: 0s - loss: 0.2387
Epoch 00020: val_loss improved from 0.19446 to 0.18305, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2386 -
val_loss: 0.1831
Epoch 21/50
258/258 [==============================] - ETA: 0s - loss: 0.2249- ETA: 0s -
loss:
Epoch 00021: val_loss improved from 0.18305 to 0.17706, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2249 -
val_loss: 0.1771
Epoch 22/50
258/258 [==============================] - ETA: 0s - loss: 0.2133
Epoch 00022: val_loss improved from 0.17706 to 0.17050, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2133 -
val_loss: 0.1705
Epoch 23/50
258/258 [==============================] - ETA: 0s - loss: 0.2013
Epoch 00023: val_loss improved from 0.17050 to 0.16339, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.2013 -
val_loss: 0.1634
Epoch 24/50
256/258 [============================>.] - ETA: 0s - loss: 0.1915
Epoch 00024: val_loss improved from 0.16339 to 0.16087, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1916 -
val_loss: 0.1609
Epoch 25/50
254/258 [============================>.] - ETA: 0s - loss: 0.1833
```

```
Epoch 00025: val_loss improved from 0.16087 to 0.15730, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1834 -
val_loss: 0.1573
Epoch 26/50
256/258 [============================>.] - ETA: 0s - loss: 0.1759
Epoch 00026: val_loss improved from 0.15730 to 0.15314, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1760 -
val_loss: 0.1531
Epoch 27/50
257/258 [============================>.] - ETA: 0s - loss: 0.1687
Epoch 00027: val_loss improved from 0.15314 to 0.15206, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1687 -
val_loss: 0.1521
Epoch 28/50
256/258 [============================>.] - ETA: 0s - loss: 0.1625
Epoch 00028: val_loss improved from 0.15206 to 0.14619, saving model to
seq2seq.h5
258/258 [==============================] - 4s 14ms/step - loss: 0.1625 -
val_loss: 0.1462
Epoch 29/50
257/258 [============================>.] - ETA: 0s - loss: 0.1574
Epoch 00029: val_loss did not improve from 0.14619
258/258 [==============================] - 4s 14ms/step - loss: 0.1574 -
val_loss: 0.1487
Epoch 30/50
256/258 [============================>.] - ETA: 0s - loss: 0.1516
Epoch 00030: val_loss improved from 0.14619 to 0.14454, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1516 -
val_loss: 0.1445
Epoch 31/50
255/258 [============================>.] - ETA: 0s - loss: 0.1473
Epoch 00031: val_loss improved from 0.14454 to 0.14326, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1472 -
val_loss: 0.1433
Epoch 32/50
254/258 [============================>.] - ETA: 0s - loss: 0.1416
Epoch 00032: val_loss improved from 0.14326 to 0.14313, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1415 -
val_loss: 0.1431
Epoch 33/50
257/258 [============================>.] - ETA: 0s - loss: 0.1382
Epoch 00033: val_loss improved from 0.14313 to 0.14277, saving model to
```

```
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1382 -
val_loss: 0.1428
Epoch 34/50
255/258 [============================>.] - ETA: 0s - loss: 0.1346
Epoch 00034: val_loss improved from 0.14277 to 0.14092, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1344 -
val_loss: 0.1409
Epoch 35/50
255/258 [============================>.] - ETA: 0s - loss: 0.1295
Epoch 00035: val_loss did not improve from 0.14092
258/258 [==============================] - 4s 14ms/step - loss: 0.1295 -
val_loss: 0.1442
Epoch 36/50
257/258 [============================>.] - ETA: 0s - loss: 0.1263
Epoch 00036: val_loss did not improve from 0.14092
258/258 [==============================] - 3s 13ms/step - loss: 0.1263 -
val_loss: 0.1426
Epoch 37/50
256/258 [============================>.] - ETA: 0s - loss: 0.1225
Epoch 00037: val_loss did not improve from 0.14092

Epoch 00037: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
258/258 [==============================] - 3s 13ms/step - loss: 0.1224 -
val_loss: 0.1424
Epoch 38/50
256/258 [============================>.] - ETA: 0s - loss: 0.1081
Epoch 00038: val_loss improved from 0.14092 to 0.13504, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1080 -
val_loss: 0.1350
Epoch 39/50
256/258 [============================>.] - ETA: 0s - loss: 0.1037
Epoch 00039: val_loss improved from 0.13504 to 0.13468, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1036 -
val_loss: 0.1347
Epoch 40/50
256/258 [============================>.] - ETA: 0s - loss: 0.1014
Epoch 00040: val_loss improved from 0.13468 to 0.13464, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.1014 -
val_loss: 0.1346
Epoch 41/50
254/258 [============================>.] - ETA: 0s - loss: 0.1003
Epoch 00041: val_loss did not improve from 0.13464
258/258 [==============================] - 3s 13ms/step - loss: 0.1003 -
```

```
val_loss: 0.1348
Epoch 42/50
255/258 [============================>.] - ETA: 0s - loss: 0.0993
Epoch 00042: val_loss did not improve from 0.13464

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
258/258 [==============================] - 3s 13ms/step - loss: 0.0994 -
val_loss: 0.1352
Epoch 43/50
257/258 [============================>.] - ETA: 0s - loss: 0.0974
Epoch 00043: val_loss improved from 0.13464 to 0.13452, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.0974 -
val_loss: 0.1345
Epoch 44/50
255/258 [============================>.] - ETA: 0s - loss: 0.0964- E
Epoch 00044: val_loss improved from 0.13452 to 0.13444, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.0964 -
val_loss: 0.1344
Epoch 45/50
257/258 [============================>.] - ETA: 0s - loss: 0.0964
Epoch 00045: val_loss improved from 0.13444 to 0.13437, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.0964 -
val_loss: 0.1344
Epoch 46/50
257/258 [============================>.] - ETA: 0s - loss: 0.0966
Epoch 00046: val_loss improved from 0.13437 to 0.13429, saving model to
seq2seq.h5
258/258 [==============================] - 3s 13ms/step - loss: 0.0965 -
val_loss: 0.1343
Epoch 47/50
258/258 [==============================] - ETA: 0s - loss: 0.0956
Epoch 00047: val_loss did not improve from 0.13429
258/258 [==============================] - 3s 13ms/step - loss: 0.0956 -
val_loss: 0.1344
Epoch 48/50
256/258 [============================>.] - ETA: 0s - loss: 0.0966
Epoch 00048: val_loss did not improve from 0.13429

Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
258/258 [==============================] - 4s 14ms/step - loss: 0.0966 -
val_loss: 0.1344
Epoch 49/50
255/258 [============================>.] - ETA: 0s - loss: 0.0958
Epoch 00049: val_loss did not improve from 0.13429
258/258 [==============================] - 4s 14ms/step - loss: 0.0958 -
```

```
val_loss: 0.1344
Epoch 50/50
254/258 [============================>.] - ETA: 0s - loss: 0.0957
Epoch 00050: val_loss did not improve from 0.13429
258/258 [==============================] - 3s 13ms/step - loss: 0.0956 -
val_loss: 0.1344
```

[20]:
```
<tensorflow.python.keras.callbacks.History at 0x298c84fd9c8>
```

[70]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, max_len, max_len)
pred_model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy')
pred_model.build(input_shape=(None, 1, max_len))
pred_model.load_weights('seq2seq.h5')
```

[71]:
```
sentence = unigram_train['input'].values[5]
print('input : ', sentence)
result = predict(sentence, unigram_vec, unigram_index_to_word, gram = 'uni')
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[5])
```

```
input :  qitted
predicted output :  quitted
actual output : quitted
```

[72]:
```
sentence = unigram_train['input'].values[11]
print('input : ', sentence)
result = predict(sentence, unigram_vec, unigram_index_to_word, gram = 'uni')
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[11])
```

```
input :  fortissiemus
predicted output :  fortissimeus
actual output : fortissimus
```

[73]:
```
sentence = unigram_train['input'].values[14]
print('input : ', sentence)
result = predict(sentence, unigram_vec, unigram_index_to_word, gram = 'uni')
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[14])
```

```
input :  numbeGr
predicted output :  number
actual output : number
```

[27]:
```
val_bleu = 0
for i in tqdm(range(val.shape[0])):
    inp = unigram_val['input'].values[i]
    out = unigram_val['output'].values[i]
```

```
    pred = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in tqdm(range(train.shape[0])):
    inp = unigram_train['input'].values[i];
    out = unigram_train['output'].values[i]
    pred = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in tqdm(range(test.shape[0])):
    inp = unigram_test['input'].values[i]
    out = unigram_test['output'].values[i]
    pred = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/unigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/unigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/unigram_test.shape[0])
```

```
100%|
  | 3678/3678 [03:16<00:00, 18.73it/s]
100%|
| 33101/33101 [29:07<00:00, 18.94it/s]
100%|
  | 3150/3150 [02:46<00:00, 18.89it/s]

BLEU Score on train:  0.8349934632437717
BLEU Score on val:  0.7469809366693739
BLEU Score on test:  0.6843990465288321
```

1.2 BiGram

[20]:
```
vocab_size = len(bigram_vec.get_vocabulary())
embedding_dim = 100
lstm_size = 256
max_len = 26
```

[28]:
```
model = Encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪max_len, max_len)
model.compile(optimizer = 'Adam',loss = 'sparse_categorical_crossentropy')

callbacks = [ModelCheckpoint('seq2seq_bigram.h5', save_best_only= True, verbose␣
 ↪= 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]
```

```python
model.fit(x = bigram_train_dataset,
          steps_per_epoch = bigram_train.shape[0]//batch_size,
          validation_data = bigram_val_dataset,
          validation_steps = bigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
6910/6910 [==============================] - ETA: 0s - loss: 0.5047
Epoch 00001: val_loss improved from inf to 0.11996, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 96s 14ms/step - loss: 0.5047 -
val_loss: 0.1200
Epoch 2/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.1247
Epoch 00002: val_loss improved from 0.11996 to 0.06599, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 100s 14ms/step - loss: 0.1246 -
val_loss: 0.0660
Epoch 3/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0858
Epoch 00003: val_loss improved from 0.06599 to 0.05437, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 104s 15ms/step - loss: 0.0858 -
val_loss: 0.0544
Epoch 4/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0722
Epoch 00004: val_loss improved from 0.05437 to 0.04924, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 104s 15ms/step - loss: 0.0722 -
val_loss: 0.0492
Epoch 5/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0649
Epoch 00005: val_loss improved from 0.04924 to 0.04526, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0649 -
val_loss: 0.0453
Epoch 6/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0601
Epoch 00006: val_loss improved from 0.04526 to 0.04325, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 106s 15ms/step - loss: 0.0601 -
val_loss: 0.0432
Epoch 7/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0566
Epoch 00007: val_loss improved from 0.04325 to 0.04161, saving model to
```

```
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0566 -
val_loss: 0.0416
Epoch 8/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0539
Epoch 00008: val_loss improved from 0.04161 to 0.04037, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 106s 15ms/step - loss: 0.0539 -
val_loss: 0.0404
Epoch 9/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0517
Epoch 00009: val_loss improved from 0.04037 to 0.03965, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0517 -
val_loss: 0.0396
Epoch 10/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0499
Epoch 00010: val_loss improved from 0.03965 to 0.03907, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 104s 15ms/step - loss: 0.0499 -
val_loss: 0.0391
Epoch 11/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0484
Epoch 00011: val_loss improved from 0.03907 to 0.03830, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 103s 15ms/step - loss: 0.0484 -
val_loss: 0.0383
Epoch 12/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0471
Epoch 00012: val_loss improved from 0.03830 to 0.03783, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0471 -
val_loss: 0.0378
Epoch 13/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0459
Epoch 00013: val_loss improved from 0.03783 to 0.03727, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 106s 15ms/step - loss: 0.0459 -
val_loss: 0.0373
Epoch 14/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0450
Epoch 00014: val_loss improved from 0.03727 to 0.03696, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0450 -
val_loss: 0.0370
Epoch 15/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0440
Epoch 00015: val_loss improved from 0.03696 to 0.03681, saving model to
```

```
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0440 -
val_loss: 0.0368
Epoch 16/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0433
Epoch 00016: val_loss improved from 0.03681 to 0.03646, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0433 -
val_loss: 0.0365
Epoch 17/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0425
Epoch 00017: val_loss improved from 0.03646 to 0.03598, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0425 -
val_loss: 0.0360
Epoch 18/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0418
Epoch 00018: val_loss improved from 0.03598 to 0.03585, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0418 -
val_loss: 0.0359
Epoch 19/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0412
Epoch 00019: val_loss improved from 0.03585 to 0.03570, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 113s 16ms/step - loss: 0.0412 -
val_loss: 0.0357
Epoch 20/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0406
Epoch 00020: val_loss improved from 0.03570 to 0.03565, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 112s 16ms/step - loss: 0.0406 -
val_loss: 0.0356
Epoch 21/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0402
Epoch 00021: val_loss improved from 0.03565 to 0.03554, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 111s 16ms/step - loss: 0.0402 -
val_loss: 0.0355
Epoch 22/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0396
Epoch 00022: val_loss improved from 0.03554 to 0.03531, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0396 -
val_loss: 0.0353
Epoch 23/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0391
Epoch 00023: val_loss improved from 0.03531 to 0.03517, saving model to
```

```
seq2seq_bigram.h5
6910/6910 [==============================] - 107s 15ms/step - loss: 0.0391 -
val_loss: 0.0352
Epoch 24/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0387
Epoch 00024: val_loss improved from 0.03517 to 0.03482, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0387 -
val_loss: 0.0348
Epoch 25/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0383
Epoch 00025: val_loss improved from 0.03482 to 0.03456, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0383 -
val_loss: 0.0346
Epoch 26/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0379
Epoch 00026: val_loss did not improve from 0.03456
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0379 -
val_loss: 0.0348
Epoch 27/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0375
Epoch 00027: val_loss improved from 0.03456 to 0.03456, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0375 -
val_loss: 0.0346
Epoch 28/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0372
Epoch 00028: val_loss improved from 0.03456 to 0.03445, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0372 -
val_loss: 0.0345
Epoch 29/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0368
Epoch 00029: val_loss did not improve from 0.03445
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0368 -
val_loss: 0.0346
Epoch 30/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0366
Epoch 00030: val_loss did not improve from 0.03445
6910/6910 [==============================] - 110s 16ms/step - loss: 0.0366 -
val_loss: 0.0346
Epoch 31/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0362
Epoch 00031: val_loss did not improve from 0.03445

Epoch 00031: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0362 -
```

```
val_loss: 0.0346
Epoch 32/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0308
Epoch 00032: val_loss improved from 0.03445 to 0.03222, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0308 -
val_loss: 0.0322
Epoch 33/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0292
Epoch 00033: val_loss improved from 0.03222 to 0.03200, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0292 -
val_loss: 0.0320
Epoch 34/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0286
Epoch 00034: val_loss improved from 0.03200 to 0.03186, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 110s 16ms/step - loss: 0.0286 -
val_loss: 0.0319
Epoch 35/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0282
Epoch 00035: val_loss improved from 0.03186 to 0.03174, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0282 -
val_loss: 0.0317
Epoch 36/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0279
Epoch 00036: val_loss improved from 0.03174 to 0.03169, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0279 -
val_loss: 0.0317
Epoch 37/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0276
Epoch 00037: val_loss did not improve from 0.03169
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0276 -
val_loss: 0.0317
Epoch 38/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0274
Epoch 00038: val_loss improved from 0.03169 to 0.03168, saving model to
seq2seq_bigram.h5

Epoch 00038: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0274 -
val_loss: 0.0317
Epoch 39/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0267
Epoch 00039: val_loss improved from 0.03168 to 0.03158, saving model to
seq2seq_bigram.h5
```

```
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0267 -
val_loss: 0.0316
Epoch 40/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0266
Epoch 00040: val_loss improved from 0.03158 to 0.03155, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 107s 15ms/step - loss: 0.0266 -
val_loss: 0.0316
Epoch 41/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0265
Epoch 00041: val_loss improved from 0.03155 to 0.03152, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 110s 16ms/step - loss: 0.0265 -
val_loss: 0.0315
Epoch 42/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0265
Epoch 00042: val_loss did not improve from 0.03152

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
6910/6910 [==============================] - 109s 16ms/step - loss: 0.0265 -
val_loss: 0.0315
Epoch 43/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0264
Epoch 00043: val_loss improved from 0.03152 to 0.03151, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0264 -
val_loss: 0.0315
Epoch 44/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0263
Epoch 00044: val_loss improved from 0.03151 to 0.03151, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 111s 16ms/step - loss: 0.0263 -
val_loss: 0.0315
Epoch 45/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0264
Epoch 00045: val_loss improved from 0.03151 to 0.03150, saving model to
seq2seq_bigram.h5

Epoch 00045: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
6910/6910 [==============================] - 112s 16ms/step - loss: 0.0264 -
val_loss: 0.0315
Epoch 46/50
6907/6910 [============================>.] - ETA: 0s - loss: 0.0263
Epoch 00046: val_loss improved from 0.03150 to 0.03149, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 107s 15ms/step - loss: 0.0263 -
val_loss: 0.0315
Epoch 47/50
```

```
6908/6910 [============================>.] - ETA: 0s - loss: 0.0263
Epoch 00047: val_loss improved from 0.03149 to 0.03149, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 106s 15ms/step - loss: 0.0263 -
val_loss: 0.0315
Epoch 48/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0264- ETA
Epoch 00048: val_loss improved from 0.03149 to 0.03149, saving model to
seq2seq_bigram.h5

Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
6910/6910 [==============================] - 105s 15ms/step - loss: 0.0264 -
val_loss: 0.0315
Epoch 49/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0264
Epoch 00049: val_loss improved from 0.03149 to 0.03149, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 107s 15ms/step - loss: 0.0264 -
val_loss: 0.0315
Epoch 50/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0263
Epoch 00050: val_loss improved from 0.03149 to 0.03149, saving model to
seq2seq_bigram.h5
6910/6910 [==============================] - 108s 16ms/step - loss: 0.0263 -
val_loss: 0.0315
```

[28]: <tensorflow.python.keras.callbacks.History at 0x224068e7048>

[21]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, max_len, max_len, bigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy')
pred_model.build(input_shape=(None, 1, max_len))
pred_model.load_weights('seq2seq_bigram.h5')
```

[42]:
```
sentence = bigram_train['input'].values[4]
print('input : ', sentence)
result = predict(sentence, bigram_vec, bigram_index_to_word, gram = 'bi')
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[4])
```

```
input :  no Lne
predicted output :  no one
actual output : no one
```

[45]:
```
sentence = bigram_train['input'].values[6]
print('input : ', sentence)
result = predict(sentence, bigram_vec, bigram_index_to_word, gram = 'bi')
print('predicted output : ',result)
```

```
print('actual output :', bigram_train['output'].values[6])
```

```
input :  for themseves
predicted output :  for themselves
actual output : for themselves
```

[46]:
```
sentence = bigram_train['input'].values[7]
print('input : ', sentence)
result = predict(sentence, bigram_vec, bigram_index_to_word, gram = 'bi')
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[7])
```

```
input :  detil of
predicted output :  detail of
actual output : detail of
```

[3]:
```
val_bleu = 0
for i in tqdm(range(bigram_val.shape[0])):
    inp = bigram_val['input'].values[i]
    out = bigram_val['output'].values[i]
    pred = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in tqdm(range(bigram_train.shape[0])):
    inp = bigram_train['input'].values[i];
    out = bigram_train['output'].values[i]
    pred = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in tqdm(range(bigram_test.shape[0])):
    inp = bigram_test['input'].values[i]
    out = bigram_test['output'].values[i]
    pred = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/bigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/bigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/bigram_test.shape[0])
```

```
BLEU Score on train:  0.9654521214897293
BLEU Score on val:  0.9467498122935679
BLEU Score on test:  0.931299275417208
```

1.3 TriGram

[50]:
```
vocab_size = len(bigram_vec.get_vocabulary())
embedding_dim = 100
```

```python
lstm_size = 256
max_len = 34
```

[52]:
```python
model = Encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪max_len, max_len)
model.compile(optimizer = 'Adam',loss = 'sparse_categorical_crossentropy')

callbacks = [ModelCheckpoint('seq2seq_trigram.h5', save_best_only= True,␣
 ↪verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = trigram_train_dataset,
          steps_per_epoch = trigram_train.shape[0]//batch_size,
          validation_data = trigram_val_dataset,
          validation_steps = trigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.7007
Epoch 00001: val_loss improved from inf to 0.21410, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 122s 18ms/step - loss: 0.7007 -
val_loss: 0.2141
Epoch 2/50
6910/6910 [==============================] - ETA: 0s - loss: 0.1938
Epoch 00002: val_loss improved from 0.21410 to 0.08815, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 129s 19ms/step - loss: 0.1938 -
val_loss: 0.0882
Epoch 3/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.1176
Epoch 00003: val_loss improved from 0.08815 to 0.06765, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 129s 19ms/step - loss: 0.1176 -
val_loss: 0.0676
Epoch 4/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0948
Epoch 00004: val_loss improved from 0.06765 to 0.05852, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 130s 19ms/step - loss: 0.0948 -
val_loss: 0.0585
Epoch 5/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0832
Epoch 00005: val_loss improved from 0.05852 to 0.05359, saving model to
seq2seq_trigram.h5
```

```
6910/6910 [==============================] - 131s 19ms/step - loss: 0.0832 -
val_loss: 0.0536
Epoch 6/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0759
Epoch 00006: val_loss improved from 0.05359 to 0.05043, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 135s 19ms/step - loss: 0.0759 -
val_loss: 0.0504
Epoch 7/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0710
Epoch 00007: val_loss improved from 0.05043 to 0.04756, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 134s 19ms/step - loss: 0.0710 -
val_loss: 0.0476
Epoch 8/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0671
Epoch 00008: val_loss improved from 0.04756 to 0.04655, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 134s 19ms/step - loss: 0.0671 -
val_loss: 0.0465
Epoch 9/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0641
Epoch 00009: val_loss improved from 0.04655 to 0.04472, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 135s 20ms/step - loss: 0.0641 -
val_loss: 0.0447
Epoch 10/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0617
Epoch 00010: val_loss improved from 0.04472 to 0.04360, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 134s 19ms/step - loss: 0.0617 -
val_loss: 0.0436
Epoch 11/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0595
Epoch 00011: val_loss improved from 0.04360 to 0.04321, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0595 -
val_loss: 0.0432
Epoch 12/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0579
Epoch 00012: val_loss improved from 0.04321 to 0.04224, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0579 -
val_loss: 0.0422
Epoch 13/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0564
Epoch 00013: val_loss improved from 0.04224 to 0.04159, saving model to
seq2seq_trigram.h5
```

```
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0564 -
val_loss: 0.0416
Epoch 14/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0550
Epoch 00014: val_loss improved from 0.04159 to 0.04106, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0550 -
val_loss: 0.0411
Epoch 15/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0538
Epoch 00015: val_loss improved from 0.04106 to 0.04039, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0538 -
val_loss: 0.0404
Epoch 16/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0528
Epoch 00016: val_loss improved from 0.04039 to 0.04007, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0528 -
val_loss: 0.0401
Epoch 17/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0519
Epoch 00017: val_loss improved from 0.04007 to 0.03975, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0519 -
val_loss: 0.0398
Epoch 18/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0510
Epoch 00018: val_loss improved from 0.03975 to 0.03891, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0510 -
val_loss: 0.0389
Epoch 19/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0501
Epoch 00019: val_loss improved from 0.03891 to 0.03879, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0501 -
val_loss: 0.0388
Epoch 20/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0494
Epoch 00020: val_loss improved from 0.03879 to 0.03860, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0494 -
val_loss: 0.0386
Epoch 21/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0488
Epoch 00021: val_loss improved from 0.03860 to 0.03802, saving model to
seq2seq_trigram.h5
```

```
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0488 -
val_loss: 0.0380
Epoch 22/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0481
Epoch 00022: val_loss improved from 0.03802 to 0.03792, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0481 -
val_loss: 0.0379
Epoch 23/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0475
Epoch 00023: val_loss improved from 0.03792 to 0.03753, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0475 -
val_loss: 0.0375
Epoch 24/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0470- ETA: 0s -
loss: 0.04
Epoch 00024: val_loss did not improve from 0.03753
6910/6910 [==============================] - 139s 20ms/step - loss: 0.0470 -
val_loss: 0.0377
Epoch 25/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0464
Epoch 00025: val_loss improved from 0.03753 to 0.03730, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 139s 20ms/step - loss: 0.0464 -
val_loss: 0.0373
Epoch 26/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0459
Epoch 00026: val_loss did not improve from 0.03730
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0459 -
val_loss: 0.0373
Epoch 27/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0455
Epoch 00027: val_loss improved from 0.03730 to 0.03691, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0455 -
val_loss: 0.0369
Epoch 28/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0450
Epoch 00028: val_loss did not improve from 0.03691
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0450 -
val_loss: 0.0370
Epoch 29/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0447
Epoch 00029: val_loss improved from 0.03691 to 0.03646, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0447 -
val_loss: 0.0365
```

```
Epoch 30/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0443
Epoch 00030: val_loss improved from 0.03646 to 0.03634, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0443 -
val_loss: 0.0363
Epoch 31/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0439
Epoch 00031: val_loss improved from 0.03634 to 0.03623, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 139s 20ms/step - loss: 0.0439 -
val_loss: 0.0362
Epoch 32/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0435
Epoch 00032: val_loss did not improve from 0.03623
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0435 -
val_loss: 0.0363
Epoch 33/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0432
Epoch 00033: val_loss improved from 0.03623 to 0.03611, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0432 -
val_loss: 0.0361
Epoch 34/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0429
Epoch 00034: val_loss improved from 0.03611 to 0.03586, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0429 -
val_loss: 0.0359
Epoch 35/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0425
Epoch 00035: val_loss did not improve from 0.03586
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0425 -
val_loss: 0.0362
Epoch 36/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0423
Epoch 00036: val_loss did not improve from 0.03586
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0423 -
val_loss: 0.0359
Epoch 37/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0420
Epoch 00037: val_loss did not improve from 0.03586

Epoch 00037: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0420 -
val_loss: 0.0359
Epoch 38/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0363
```

```
Epoch 00038: val_loss improved from 0.03586 to 0.03275, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0363 -
val_loss: 0.0327
Epoch 39/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0346
Epoch 00039: val_loss improved from 0.03275 to 0.03238, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 137s 20ms/step - loss: 0.0346 -
val_loss: 0.0324
Epoch 40/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0339
Epoch 00040: val_loss improved from 0.03238 to 0.03222, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0339 -
val_loss: 0.0322
Epoch 41/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0336
Epoch 00041: val_loss improved from 0.03222 to 0.03208, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0336 -
val_loss: 0.0321
Epoch 42/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0333
Epoch 00042: val_loss improved from 0.03208 to 0.03205, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0333 -
val_loss: 0.0321
Epoch 43/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0331
Epoch 00043: val_loss did not improve from 0.03205
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0331 -
val_loss: 0.0321
Epoch 44/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0329
Epoch 00044: val_loss improved from 0.03205 to 0.03197, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 139s 20ms/step - loss: 0.0329 -
val_loss: 0.0320
Epoch 45/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0327
Epoch 00045: val_loss improved from 0.03197 to 0.03190, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 138s 20ms/step - loss: 0.0327 -
val_loss: 0.0319
Epoch 46/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0326
Epoch 00046: val_loss did not improve from 0.03190
```

```
6910/6910 [==============================] - 134s 19ms/step - loss: 0.0326 -
val_loss: 0.0319
Epoch 47/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0324
Epoch 00047: val_loss did not improve from 0.03190

Epoch 00047: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 135s 20ms/step - loss: 0.0324 -
val_loss: 0.0319
Epoch 48/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0318
Epoch 00048: val_loss improved from 0.03190 to 0.03173, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 136s 20ms/step - loss: 0.0318 -
val_loss: 0.0317
Epoch 49/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0316
Epoch 00049: val_loss improved from 0.03173 to 0.03169, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 134s 19ms/step - loss: 0.0316 -
val_loss: 0.0317
Epoch 50/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0315
Epoch 00050: val_loss improved from 0.03169 to 0.03166, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 133s 19ms/step - loss: 0.0315 -
val_loss: 0.0317
```

[52]: <tensorflow.python.keras.callbacks.History at 0x224099166c8>

[59]:
```
from tensorflow.keras.optimizers import Adam
model = Encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪max_len, max_len)
model.compile(optimizer = Adam(1.0000000474974514e-05),loss =␣
 ↪'sparse_categorical_crossentropy')

callbacks = [ModelCheckpoint('seq2seq_trigram.h5', save_best_only= True,␣
 ↪verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.build(input_shape = (None, batch_size, max_len))
model.load_weights('seq2seq_trigram.h5')

model.fit(x = trigram_train_dataset,
          steps_per_epoch = trigram_train.shape[0]//batch_size,
          validation_data = trigram_val_dataset,
```

```
          validation_steps = trigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0315
Epoch 00001: val_loss improved from inf to 0.03166, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 123s 18ms/step - loss: 0.0315 -
val_loss: 0.0317
Epoch 2/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00002: val_loss improved from 0.03166 to 0.03165, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 128s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 3/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0314
Epoch 00003: val_loss did not improve from 0.03165
6910/6910 [==============================] - 130s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 4/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0315
Epoch 00004: val_loss improved from 0.03165 to 0.03163, saving model to
seq2seq_trigram.h5

Epoch 00004: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
6910/6910 [==============================] - 131s 19ms/step - loss: 0.0315 -
val_loss: 0.0316
Epoch 5/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00005: val_loss improved from 0.03163 to 0.03161, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 132s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 6/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0313- ETA: 0s -
loss: 0.0
Epoch 00006: val_loss improved from 0.03161 to 0.03160, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 130s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 7/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00007: val_loss improved from 0.03160 to 0.03159, saving model to
seq2seq_trigram.h5
```

```
Epoch 00007: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
6910/6910 [==============================] - 132s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 8/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00008: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 133s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 9/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0313
Epoch 00009: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 131s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 10/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0313
Epoch 00010: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5

Epoch 00010: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
6910/6910 [==============================] - 130s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 11/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0314
Epoch 00011: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 133s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 12/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00012: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 133s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 13/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00013: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5

Epoch 00013: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-09.
6910/6910 [==============================] - 131s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 14/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00014: val_loss did not improve from 0.03159
6910/6910 [==============================] - 132s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
```

```
Epoch 15/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00015: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 130s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 16/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0313
Epoch 00016: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5

Epoch 00016: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-10.
6910/6910 [==============================] - 132s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 17/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0313
Epoch 00017: val_loss improved from 0.03159 to 0.03159, saving model to
seq2seq_trigram.h5
6910/6910 [==============================] - 127s 18ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 18/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00018: val_loss did not improve from 0.03159
6910/6910 [==============================] - 127s 18ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 19/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0313
Epoch 00019: val_loss did not improve from 0.03159

Epoch 00019: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-11.
6910/6910 [==============================] - 128s 19ms/step - loss: 0.0313 -
val_loss: 0.0316
Epoch 20/50
6908/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00020: val_loss did not improve from 0.03159
6910/6910 [==============================] - 128s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 21/50
6909/6910 [============================>.] - ETA: 0s - loss: 0.0314
Epoch 00021: val_loss did not improve from 0.03159
6910/6910 [==============================] - 128s 19ms/step - loss: 0.0314 -
val_loss: 0.0316
Epoch 22/50
6910/6910 [==============================] - ETA: 0s - loss: 0.0313
Epoch 00022: val_loss did not improve from 0.03159

Epoch 00022: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-12.
6910/6910 [==============================] - 129s 19ms/step - loss: 0.0313 -
```

```
val_loss: 0.0316
Epoch 00022: early stopping
```

[59]: <tensorflow.python.keras.callbacks.History at 0x224047a6788>

[67]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, max_len, max_len, trigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy')
pred_model.build(input_shape=(None, 1, max_len))
pred_model.load_weights('seq2seq_trigram.h5')
```

[68]:
```
sentence = trigram_train['input'].values[4]
print('input : ', sentence)
result = predict(sentence, trigram_vec, trigram_index_to_word, gram = 'tri')
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[4])
```

```
input :  woman oAf him
predicted output :  woman of him
actual output : woman of him
```

[72]:
```
sentence = trigram_train['input'].values[7]
print('input : ', sentence)
result = predict(sentence, trigram_vec, trigram_index_to_word, gram = 'tri')
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[7])
```

```
input :  endurXe a collision
predicted output :  endure a collision
actual output : endure a collision
```

[75]:
```
sentence = trigram_train['input'].values[10]
print('input : ', sentence)
result = predict(sentence, trigram_vec, trigram_index_to_word, gram = 'tri')
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[10])
```

```
input :  marshals sitti g on
predicted output :  marshals sitting on
actual output : marshals sitting on
```

[4]:
```
val_bleu = 0
for i in tqdm(range(trigram_val.shape[0])):
    inp = trigram_val['input'].values[i]
    out = trigram_val['output'].values[i]
    pred = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
```

```
for i in tqdm(range(train.shape[0])):
    inp = trigram_train['input'].values[i];
    out = trigram_train['output'].values[i]
    pred = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in tqdm(range(test.shape[0])):
    inp = trigram_test['input'].values[i]
    out = trigram_test['output'].values[i]
    pred = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/trigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/trigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/trigram_test.shape[0])
```

```
BLEU Score on train:  0.9684256422311769
BLEU Score on val:  0.9575594555349892
BLEU Score on test:  0.949065764861953
```

2. Seq2Seq with Attention Mechanism

[77]:
```
class Encoder(tf.keras.layers.Layer):
    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):
        super(Encoder, self).__init__()
        self.lstm_size = lstm_size
        #Initialize Embedding layer
        self.enc_embed = Embedding(input_dim = inp_vocab_size, output_dim =␣
↪embedding_size)
        #Intialize Encoder LSTM layer
        self.enc_lstm = LSTM(lstm_size, return_sequences = True, return_state =␣
↪True, dropout = 0.4)

    def call(self,input_sequence,states):
        embedding = self.enc_embed(input_sequence)
        output_state, enc_h, enc_c = self.enc_lstm(embedding, initial_state =␣
↪states)
        return output_state, enc_h, enc_c

    def initialize_states(self,batch_size):
        return [tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size,␣
↪self.lstm_size))]

class Attention(tf.keras.layers.Layer):

    def __init__(self,scoring_function, att_units):
        super(Attention, self).__init__()
```

```
        self.scoring_function = scoring_function
        if scoring_function == 'dot':
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'general':
          # Intialize variables needed for General score function here
            self.W = Dense(att_units)
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'concat':
          # Intialize variables needed for Concat score function here
            self.W1 = Dense(att_units)
            self.W2 = Dense(att_units)
            self.V = Dense(1)
    def call(self,decoder_hidden_state,encoder_output):

        decoder_hidden_state = tf.expand_dims(decoder_hidden_state, 1)

        if self.scoring_function == 'dot':
            # Implement Dot score function here
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state,␣
↪(0, 2, 1)), encoder_output]), (0, 2,1))

        elif self.scoring_function == 'general':
            # Implement General score function here
            mul = self.W(encoder_output)
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state,␣
↪(0, 2, 1)), mul]), (0, 2,1))

        elif self.scoring_function == 'concat':
            # Implement General score function here
            inter = self.W1(decoder_hidden_state) + self.W2(encoder_output)
            tan = tf.nn.tanh(inter)
            score = self.V(tan)
        attention_weights = tf.nn.softmax(score, axis =1)
        context_vector = attention_weights * encoder_output
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights

class OneStepDecoder(tf.keras.layers.Layer):
    def __init__(self,tar_vocab_size, embedding_dim, input_length, dec_units␣
↪,score_fun ,att_units):
        super(OneStepDecoder, self).__init__()
      # Initialize decoder embedding layer, LSTM and any other objects needed
        self.embed_dec = Embedding(input_dim = tar_vocab_size, output_dim =␣
↪embedding_dim)
        self.lstm = LSTM(dec_units, return_sequences = True, return_state =␣
↪True, dropout = 0.4)
```

```
        self.attention = Attention(scoring_function = score_fun, att_units =␣
↪att_units)
        self.fc = Dense(tar_vocab_size)

    def call(self,input_to_decoder, encoder_output, state_h,state_c):
        embed = self.embed_dec(input_to_decoder)
        context_vect, attention_weights = self.attention(state_h,␣
↪encoder_output)
        final_inp = tf.concat([tf.expand_dims(context_vect, 1), embed], axis =␣
↪-1)
        out, dec_h, dec_c = self.lstm(final_inp, [state_h, state_c])
        out = tf.reshape(out, (-1, out.shape[2]))
        output = self.fc(out)
        output = Dropout(0.5)(output)
        return output, dec_h, dec_c, attention_weights, context_vect

class encoder_decoder(tf.keras.Model):
    def __init__(self, inp_vocab_size, out_vocab_size, embedding_dim,␣
↪enc_units, dec_units, max_len_inp, max_len_out, score_fun, att_units,␣
↪batch_size):
        #Intialize objects from encoder decoder
        super(encoder_decoder, self).__init__()
        self.encoder = Encoder(inp_vocab_size, embedding_dim, enc_units,␣
↪max_len_inp)
        self.one_step_decoder = OneStepDecoder(out_vocab_size, embedding_dim,␣
↪max_len_out, dec_units ,score_fun ,att_units)
        self.batch_size = batch_size

    def call(self, data):
        enc_inp, dec_inp = data[0], data[1]
        initial_state = self.encoder.initialize_states(self.batch_size)
        enc_output, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        all_outputs = tf.TensorArray(dtype = tf.float32, size= max_len)

        dec_h = enc_h
        dec_c = enc_c
        for timestep in range(max_len):
            # Call onestepdecoder for each token in decoder_input
            output, dec_h, dec_c, _, _ = self.one_step_decoder(dec_inp[:,␣
↪timestep:timestep+1],
                                                               enc_output,
                                                               dec_h,
                                                               dec_c)
            # Store the output in tensorarray
            all_outputs = all_outputs.write(timestep, output)
        # Return the tensor array
```

```
        all_outputs = tf.transpose(all_outputs.stack(), (1, 0, 2))
        # return the decoder output
        return all_outputs

class pred_Encoder_decoder(tf.keras.Model):
    def __init__(self, inp_vocab_size, out_vocab_size, embedding_dim,␣
↪enc_units, dec_units, max_len_ita, max_len_eng, score_fun, att_units,␣
↪word_to_index):
        #Intialize objects from encoder decoder
        super(pred_Encoder_decoder, self).__init__()
        self.encoder = Encoder(inp_vocab_size, embedding_dim, enc_units,␣
↪max_len_ita)
        self.one_step_decoder = OneStepDecoder(out_vocab_size, embedding_dim,␣
↪max_len_eng, dec_units ,score_fun ,att_units)
        self.batch_size = batch_size
        self.word_to_index = word_to_index

    def call(self, params):
        enc_inp = params[0]
        initial_state = self.encoder.initialize_states(1)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        pred = tf.expand_dims([self.word_to_index['<SOW>']], 0)
        dec_h = enc_h
        dec_c = enc_c
        all_pred = []
        all_attention = []
        for t in range(max_len):
            output, dec_h,dec_c, attention, _ = self.one_step_decoder(pred,␣
↪output_state, dec_h, dec_c)
            pred = tf.argmax(output, axis = -1)
            all_pred.append(pred)
            pred = tf.expand_dims(pred, 0)
            all_attention.append(attention)
        return all_pred, all_attention
```

[78]:
```
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_)
```

[79]:
```
def predict(seq, vectorizer, index_to_word, gram = 'uni'):
    if gram =='uni':
        seq = '<SOW> '+' '.join(list(seq))+' <EOW>'
```

```
    else:
        seq = '<SOW>*'+'*'.join(list(seq))+'*<EOW>'
    seq = vectorizer([seq])
    pred, attention_weights = pred_model.predict(tf.expand_dims(seq, 0))
    output = []
    for i in pred:
        word = index_to_word[i[0]]
        if word == '<EOW>':
            break
        output.append(word)
    return ''.join(output), np.squeeze(np.squeeze(np.array(attention_weights),␣
↪1), -1)
```

[80]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

def plot_attention(attention, sentence, predicted_sentence):
    fig = plt.figure(figsize=(10,10))
    ax = fig.add_subplot(1, 1, 1)
    ax.matshow(attention, cmap='viridis')

    fontdict = {'fontsize': 14}

    ax.set_xticklabels([''] + sentence, fontdict=fontdict, rotation=90)
    ax.set_yticklabels([''] + predicted_sentence, fontdict=fontdict)

    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

    plt.show()
```

2.1 UniGram

[76]:
```
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 22
```

[31]:
```
model = encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
↪lstm_size, maxlen, maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)

callbacks = [ModelCheckpoint('Attention_concat_lstm.h5', save_best_only= True,␣
↪verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]
```

```
model.fit(x = unigram_train_dataset,
          steps_per_epoch = unigram_train.shape[0]//batch_size,
          validation_data = unigram_val_dataset,
          validation_steps = unigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
258/258 [==============================] - ETA: 0s - loss: 1.3239
Epoch 00001: val_loss improved from inf to 0.96448, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 19s 74ms/step - loss: 1.3239 -
val_loss: 0.9645
Epoch 2/50
258/258 [==============================] - ETA: 0s - loss: 1.1640
Epoch 00002: val_loss improved from 0.96448 to 0.73190, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 13s 50ms/step - loss: 1.1640 -
val_loss: 0.7319
Epoch 3/50
257/258 [============================>.] - ETA: 0s - loss: 1.0338
Epoch 00003: val_loss improved from 0.73190 to 0.44536, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 13s 51ms/step - loss: 1.0337 -
val_loss: 0.4454
Epoch 4/50
258/258 [==============================] - ETA: 0s - loss: 0.8934- ETA:
Epoch 00004: val_loss improved from 0.44536 to 0.26591, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 13s 51ms/step - loss: 0.8934 -
val_loss: 0.2659
Epoch 5/50
258/258 [==============================] - ETA: 0s - loss: 0.8349
Epoch 00005: val_loss improved from 0.26591 to 0.21211, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 13s 51ms/step - loss: 0.8349 -
val_loss: 0.2121
Epoch 6/50
258/258 [==============================] - ETA: 0s - loss: 0.8071
Epoch 00006: val_loss improved from 0.21211 to 0.18196, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.8071 -
val_loss: 0.1820
Epoch 7/50
258/258 [==============================] - ETA: 0s - loss: 0.7956
Epoch 00007: val_loss improved from 0.18196 to 0.16000, saving model to
Attention_concat_lstm.h5
```

```
258/258 [==============================] - 13s 52ms/step - loss: 0.7956 -
val_loss: 0.1600
Epoch 8/50
257/258 [============================>.] - ETA: 0s - loss: 0.7887- ETA: 0s - l
Epoch 00008: val_loss improved from 0.16000 to 0.15305, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 52ms/step - loss: 0.7887 -
val_loss: 0.1530
Epoch 9/50
258/258 [==============================] - ETA: 0s - loss: 0.7819
Epoch 00009: val_loss improved from 0.15305 to 0.14978, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7819 -
val_loss: 0.1498
Epoch 10/50
257/258 [============================>.] - ETA: 0s - loss: 0.7781
Epoch 00010: val_loss improved from 0.14978 to 0.14204, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7782 -
val_loss: 0.1420
Epoch 11/50
258/258 [==============================] - ETA: 0s - loss: 0.7725
Epoch 00011: val_loss improved from 0.14204 to 0.13989, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7725 -
val_loss: 0.1399
Epoch 12/50
258/258 [==============================] - ETA: 0s - loss: 0.7676
Epoch 00012: val_loss improved from 0.13989 to 0.13404, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7676 -
val_loss: 0.1340
Epoch 13/50
258/258 [==============================] - ETA: 0s - loss: 0.7644
Epoch 00013: val_loss improved from 0.13404 to 0.12980, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7644 -
val_loss: 0.1298
Epoch 14/50
258/258 [==============================] - ETA: 0s - loss: 0.7605
Epoch 00014: val_loss improved from 0.12980 to 0.12613, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7605 -
val_loss: 0.1261
Epoch 15/50
258/258 [==============================] - ETA: 0s - loss: 0.7610
Epoch 00015: val_loss did not improve from 0.12613
258/258 [==============================] - 14s 53ms/step - loss: 0.7610 -
```

```
val_loss: 0.1295
Epoch 16/50
258/258 [==============================] - ETA: 0s - loss: 0.7585
Epoch 00016: val_loss did not improve from 0.12613
258/258 [==============================] - 14s 53ms/step - loss: 0.7585 -
val_loss: 0.1265
Epoch 17/50
258/258 [==============================] - ETA: 0s - loss: 0.7560
Epoch 00017: val_loss improved from 0.12613 to 0.12149, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7560 -
val_loss: 0.1215
Epoch 18/50
258/258 [==============================] - ETA: 0s - loss: 0.7552
Epoch 00018: val_loss improved from 0.12149 to 0.12073, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 55ms/step - loss: 0.7552 -
val_loss: 0.1207
Epoch 19/50
258/258 [==============================] - ETA: 0s - loss: 0.7534
Epoch 00019: val_loss improved from 0.12073 to 0.11933, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 55ms/step - loss: 0.7534 -
val_loss: 0.1193
Epoch 20/50
258/258 [==============================] - ETA: 0s - loss: 0.7519
Epoch 00020: val_loss improved from 0.11933 to 0.11756, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7519 -
val_loss: 0.1176
Epoch 21/50
257/258 [============================>.] - ETA: 0s - loss: 0.7455
Epoch 00021: val_loss improved from 0.11756 to 0.11589, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7454 -
val_loss: 0.1159
Epoch 22/50
258/258 [==============================] - ETA: 0s - loss: 0.7493
Epoch 00022: val_loss did not improve from 0.11589
258/258 [==============================] - 14s 52ms/step - loss: 0.7493 -
val_loss: 0.1170
Epoch 23/50
258/258 [==============================] - ETA: 0s - loss: 0.7449
Epoch 00023: val_loss improved from 0.11589 to 0.11506, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 55ms/step - loss: 0.7449 -
val_loss: 0.1151
Epoch 24/50
```

```
258/258 [==============================] - ETA: 0s - loss: 0.7447
Epoch 00024: val_loss did not improve from 0.11506
258/258 [==============================] - 14s 53ms/step - loss: 0.7447 -
val_loss: 0.1168
Epoch 25/50
257/258 [============================>.] - ETA: 0s - loss: 0.7438
Epoch 00025: val_loss did not improve from 0.11506
258/258 [==============================] - 14s 53ms/step - loss: 0.7440 -
val_loss: 0.1155
Epoch 26/50
258/258 [==============================] - ETA: 0s - loss: 0.7442
Epoch 00026: val_loss improved from 0.11506 to 0.11340, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7442 -
val_loss: 0.1134
Epoch 27/50
258/258 [==============================] - ETA: 0s - loss: 0.7413
Epoch 00027: val_loss improved from 0.11340 to 0.11310, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7413 -
val_loss: 0.1131
Epoch 28/50
258/258 [==============================] - ETA: 0s - loss: 0.7421
Epoch 00028: val_loss improved from 0.11310 to 0.11093, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7421 -
val_loss: 0.1109
Epoch 29/50
258/258 [==============================] - ETA: 0s - loss: 0.7379
Epoch 00029: val_loss improved from 0.11093 to 0.10953, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7379 -
val_loss: 0.1095
Epoch 30/50
258/258 [==============================] - ETA: 0s - loss: 0.7356
Epoch 00030: val_loss did not improve from 0.10953
258/258 [==============================] - 14s 53ms/step - loss: 0.7356 -
val_loss: 0.1112
Epoch 31/50
257/258 [============================>.] - ETA: 0s - loss: 0.7349
Epoch 00031: val_loss improved from 0.10953 to 0.10774, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7350 -
val_loss: 0.1077
Epoch 32/50
258/258 [==============================] - ETA: 0s - loss: 0.7343
Epoch 00032: val_loss improved from 0.10774 to 0.10720, saving model to
Attention_concat_lstm.h5
```

```
258/258 [==============================] - 14s 53ms/step - loss: 0.7343 -
val_loss: 0.1072
Epoch 33/50
258/258 [==============================] - ETA: 0s - loss: 0.7364
Epoch 00033: val_loss did not improve from 0.10720
258/258 [==============================] - 14s 53ms/step - loss: 0.7364 -
val_loss: 0.1104
Epoch 34/50
257/258 [============================>.] - ETA: 0s - loss: 0.7355
Epoch 00034: val_loss improved from 0.10720 to 0.10713, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7355 -
val_loss: 0.1071
Epoch 35/50
257/258 [============================>.] - ETA: 0s - loss: 0.7310
Epoch 00035: val_loss did not improve from 0.10713

Epoch 00035: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
258/258 [==============================] - 14s 53ms/step - loss: 0.7311 -
val_loss: 0.1073
Epoch 36/50
257/258 [============================>.] - ETA: 0s - loss: 0.7241
Epoch 00036: val_loss improved from 0.10713 to 0.10300, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7241 -
val_loss: 0.1030
Epoch 37/50
258/258 [==============================] - ETA: 0s - loss: 0.7243
Epoch 00037: val_loss improved from 0.10300 to 0.10202, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7243 -
val_loss: 0.1020
Epoch 38/50
258/258 [==============================] - ETA: 0s - loss: 0.7227
Epoch 00038: val_loss improved from 0.10202 to 0.10193, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7227 -
val_loss: 0.1019
Epoch 39/50
258/258 [==============================] - ETA: 0s - loss: 0.7210
Epoch 00039: val_loss improved from 0.10193 to 0.10119, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7210 -
val_loss: 0.1012
Epoch 40/50
258/258 [==============================] - ETA: 0s - loss: 0.7227
Epoch 00040: val_loss did not improve from 0.10119
258/258 [==============================] - 14s 54ms/step - loss: 0.7227 -
```

```
val_loss: 0.1014
Epoch 41/50
257/258 [============================>.] - ETA: 0s - loss: 0.7211
Epoch 00041: val_loss improved from 0.10119 to 0.10118, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7213 -
val_loss: 0.1012
Epoch 42/50
258/258 [==============================] - ETA: 0s - loss: 0.7175
Epoch 00042: val_loss did not improve from 0.10118

Epoch 00042: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
258/258 [==============================] - 14s 54ms/step - loss: 0.7175 -
val_loss: 0.1013
Epoch 43/50
258/258 [==============================] - ETA: 0s - loss: 0.7193
Epoch 00043: val_loss improved from 0.10118 to 0.10111, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7193 -
val_loss: 0.1011
Epoch 44/50
258/258 [==============================] - ETA: 0s - loss: 0.7186
Epoch 00044: val_loss improved from 0.10111 to 0.10101, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7186 -
val_loss: 0.1010
Epoch 45/50
258/258 [==============================] - ETA: 0s - loss: 0.7193
Epoch 00045: val_loss improved from 0.10101 to 0.10088, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 54ms/step - loss: 0.7193 -
val_loss: 0.1009
Epoch 46/50
258/258 [==============================] - ETA: 0s - loss: 0.7194
Epoch 00046: val_loss improved from 0.10088 to 0.10086, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7194 -
val_loss: 0.1009
Epoch 47/50
258/258 [==============================] - ETA: 0s - loss: 0.7210
Epoch 00047: val_loss improved from 0.10086 to 0.10083, saving model to
Attention_concat_lstm.h5
258/258 [==============================] - 14s 53ms/step - loss: 0.7210 -
val_loss: 0.1008
Epoch 48/50
258/258 [==============================] - ETA: 0s - loss: 0.7186
Epoch 00048: val_loss did not improve from 0.10083
```

```
Epoch 00048: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
258/258 [==============================] - 14s 53ms/step - loss: 0.7186 -
val_loss: 0.1009
Epoch 49/50
257/258 [============================>.] - ETA: 0s - loss: 0.7182
Epoch 00049: val_loss did not improve from 0.10083
258/258 [==============================] - 14s 54ms/step - loss: 0.7182 -
val_loss: 0.1009
Epoch 50/50
258/258 [==============================] - ETA: 0s - loss: 0.7206
Epoch 00050: val_loss did not improve from 0.10083
258/258 [==============================] - 14s 54ms/step - loss: 0.7206 -
val_loss: 0.1008
```

[31]:
```
<tensorflow.python.keras.callbacks.History at 0x298c951fd88>
```

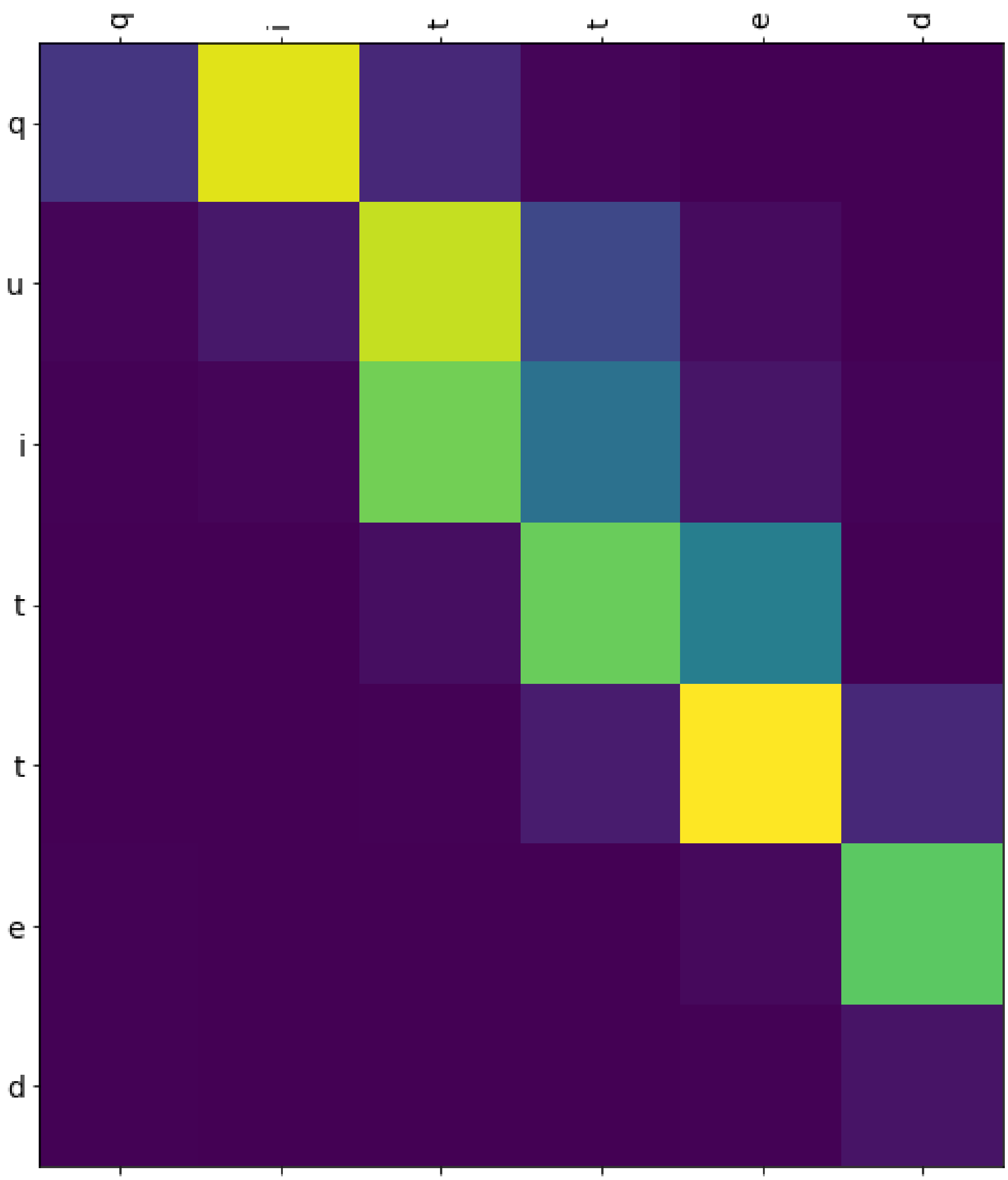
[78]:
```python
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, lstm_size, maxlen, maxlen, 'concat', att_units)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('Attention_concat_lstm.h5')
```

[81]:
```python
sentence = unigram_train['input'].values[5]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
 ↪gram = 'uni')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[5])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  qitted
predicted output :  quitted
actual output : quitted
```

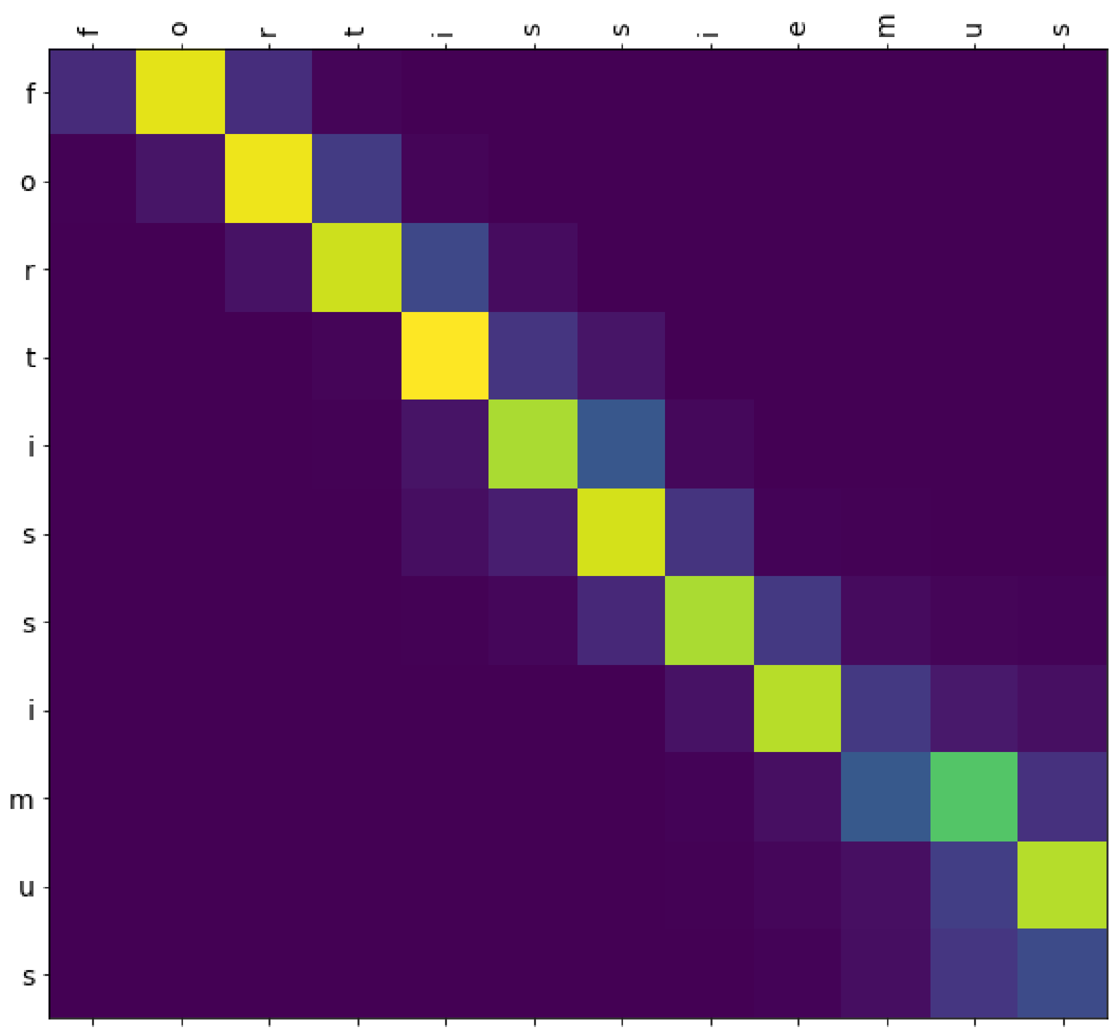


[82]:
```
sentence = unigram_train['input'].values[11]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
 ↪gram = 'uni')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[11])
```

```
attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  fortissiemus
predicted output :  fortissimus
actual output : fortissimus
```

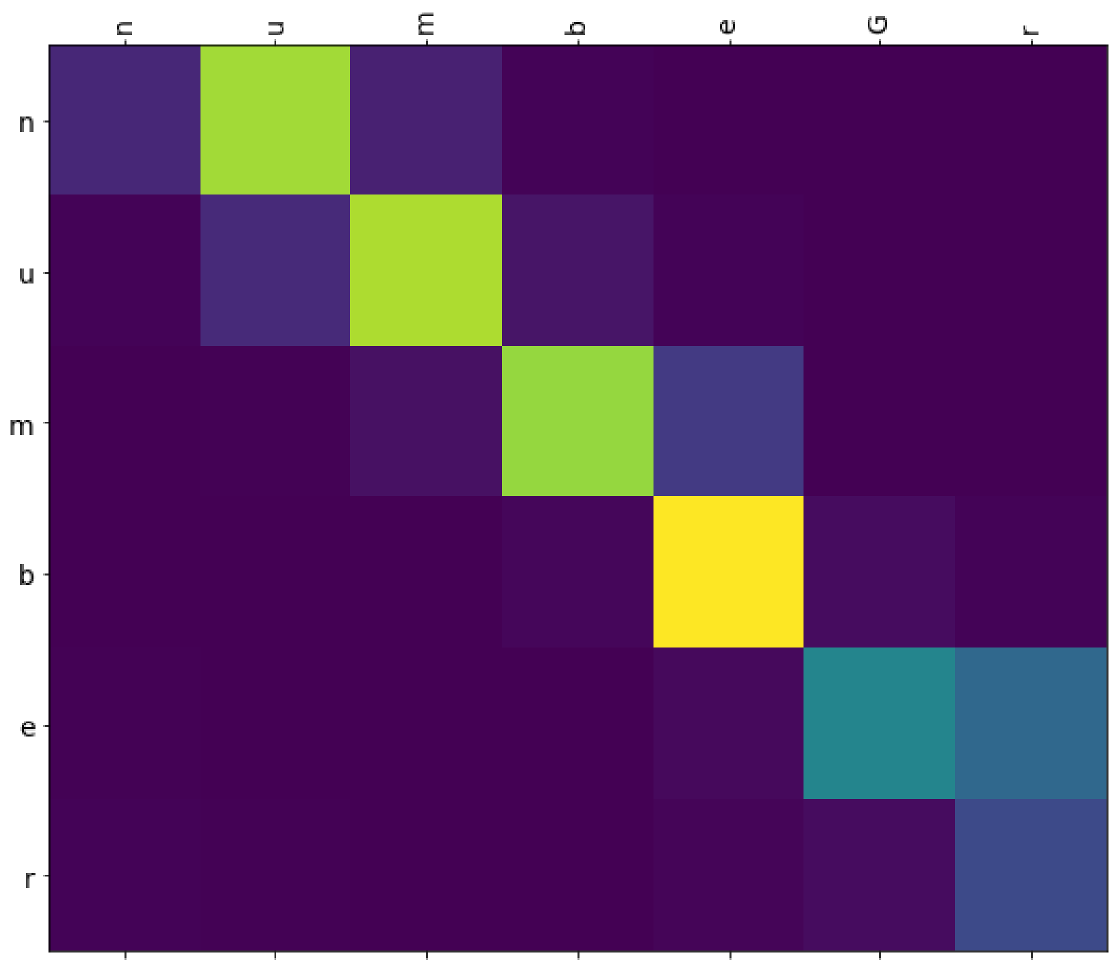


[83]:
```
sentence = unigram_train['input'].values[14]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
 ↪gram = 'uni')

print('input:', sentence)
print('predicted output:',result)
print('actual output:', unigram_train['output'].values[14])
```

```
attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input: numbeGr
predicted output: number
actual output: number
```



[39]:
```
val_bleu = 0
for i in tqdm(range(unigram_val.shape[0])):
    inp = unigram_val['input'].values[i]
    out = unigram_val['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in tqdm(range(unigram_train.shape[0])):
    inp = unigram_train['input'].values[i];
```

```
    out = unigram_train['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in tqdm(range(unigram_test.shape[0])):
    inp = unigram_test['input'].values[i]
    out = unigram_test['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/unigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/unigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/unigram_test.shape[0])
```

```
100%|
  | 3678/3678 [03:40<00:00, 16.65it/s]
100%|
 | 33101/33101 [32:52<00:00, 16.78it/s]
100%|
  | 3150/3150 [03:06<00:00, 16.88it/s]

BLEU Score on train:  0.8698739128364643
BLEU Score on val:  0.792070857805782
BLEU Score on test:  0.7071433929743683
```

2.2 BiGram

[83]:
```
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 26
```

[85]:
```
model = encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪lstm_size, maxlen, maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)

callbacks = [ModelCheckpoint('Attention_concat_lstm_bigram.h5', save_best_only=␣
 ↪True, verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = bigram_train_dataset,
          steps_per_epoch = bigram_train.shape[0]//batch_size,
          validation_data = bigram_val_dataset,
          validation_steps = bigram_val.shape[0]//batch_size,
          epochs = 100,
```

```
            verbose = 1,
            callbacks = callbacks)
```

```
Epoch 1/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8288
Epoch 00001: val_loss improved from inf to 0.06538, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 471s 68ms/step - loss: 0.8288 -
val_loss: 0.0654
Epoch 2/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7482
Epoch 00002: val_loss improved from 0.06538 to 0.05269, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 486s 70ms/step - loss: 0.7482 -
val_loss: 0.0527
Epoch 3/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7420
Epoch 00003: val_loss improved from 0.05269 to 0.04566, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 498s 72ms/step - loss: 0.7420 -
val_loss: 0.0457
Epoch 4/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7379
Epoch 00004: val_loss improved from 0.04566 to 0.04211, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 474s 69ms/step - loss: 0.7379 -
val_loss: 0.0421
Epoch 5/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7357
Epoch 00005: val_loss improved from 0.04211 to 0.03942, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 463s 67ms/step - loss: 0.7357 -
val_loss: 0.0394
Epoch 6/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7333
Epoch 00006: val_loss improved from 0.03942 to 0.03707, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 467s 68ms/step - loss: 0.7333 -
val_loss: 0.0371
Epoch 7/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7320
Epoch 00007: val_loss improved from 0.03707 to 0.03556, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 486s 70ms/step - loss: 0.7320 -
val_loss: 0.0356
Epoch 8/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7315
Epoch 00008: val_loss improved from 0.03556 to 0.03482, saving model to
```

```
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 472s 68ms/step - loss: 0.7315 -
val_loss: 0.0348
Epoch 9/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7305
Epoch 00009: val_loss improved from 0.03482 to 0.03351, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 461s 67ms/step - loss: 0.7305 -
val_loss: 0.0335
Epoch 10/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7293
Epoch 00010: val_loss improved from 0.03351 to 0.03246, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7293 -
val_loss: 0.0325
Epoch 11/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7283
Epoch 00011: val_loss improved from 0.03246 to 0.03200, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 468s 68ms/step - loss: 0.7283 -
val_loss: 0.0320
Epoch 12/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7279
Epoch 00012: val_loss improved from 0.03200 to 0.03148, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7279 -
val_loss: 0.0315
Epoch 13/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7278
Epoch 00013: val_loss improved from 0.03148 to 0.03059, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 461s 67ms/step - loss: 0.7278 -
val_loss: 0.0306
Epoch 14/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7273
Epoch 00014: val_loss improved from 0.03059 to 0.02997, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 465s 67ms/step - loss: 0.7273 -
val_loss: 0.0300
Epoch 15/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7262
Epoch 00015: val_loss improved from 0.02997 to 0.02988, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 468s 68ms/step - loss: 0.7262 -
val_loss: 0.0299
Epoch 16/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7260
Epoch 00016: val_loss improved from 0.02988 to 0.02935, saving model to
```

```
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 466s 67ms/step - loss: 0.7260 -
val_loss: 0.0294
Epoch 17/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7258
Epoch 00017: val_loss improved from 0.02935 to 0.02916, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 469s 68ms/step - loss: 0.7258 -
val_loss: 0.0292
Epoch 18/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7256
Epoch 00018: val_loss improved from 0.02916 to 0.02896, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 478s 69ms/step - loss: 0.7256 -
val_loss: 0.0290
Epoch 19/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7254- ETA:
Epoch 00019: val_loss improved from 0.02896 to 0.02854, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 477s 69ms/step - loss: 0.7254 -
val_loss: 0.0285
Epoch 20/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7250
Epoch 00020: val_loss improved from 0.02854 to 0.02826, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 485s 70ms/step - loss: 0.7250 -
val_loss: 0.0283
Epoch 21/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7245
Epoch 00021: val_loss improved from 0.02826 to 0.02785, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 483s 70ms/step - loss: 0.7245 -
val_loss: 0.0278
Epoch 22/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7247
Epoch 00022: val_loss did not improve from 0.02785
6910/6910 [==============================] - 473s 69ms/step - loss: 0.7247 -
val_loss: 0.0288
Epoch 23/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7239
Epoch 00023: val_loss did not improve from 0.02785
6910/6910 [==============================] - 467s 68ms/step - loss: 0.7239 -
val_loss: 0.0279
Epoch 24/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7236
Epoch 00024: val_loss improved from 0.02785 to 0.02764, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7236 -
```

```
val_loss: 0.0276
Epoch 25/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7239
Epoch 00025: val_loss improved from 0.02764 to 0.02751, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 467s 68ms/step - loss: 0.7239 -
val_loss: 0.0275
Epoch 26/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7238
Epoch 00026: val_loss did not improve from 0.02751
6910/6910 [==============================] - 465s 67ms/step - loss: 0.7238 -
val_loss: 0.0277
Epoch 27/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7236
Epoch 00027: val_loss improved from 0.02751 to 0.02704, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 473s 69ms/step - loss: 0.7236 -
val_loss: 0.0270
Epoch 28/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7231
Epoch 00028: val_loss did not improve from 0.02704
6910/6910 [==============================] - 464s 67ms/step - loss: 0.7231 -
val_loss: 0.0271
Epoch 29/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7233
Epoch 00029: val_loss improved from 0.02704 to 0.02701, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 680s 98ms/step - loss: 0.7233 -
val_loss: 0.0270
Epoch 30/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7232
Epoch 00030: val_loss improved from 0.02701 to 0.02620, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7232 -
val_loss: 0.0262
Epoch 31/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7231
Epoch 00031: val_loss did not improve from 0.02620
6910/6910 [==============================] - 472s 68ms/step - loss: 0.7231 -
val_loss: 0.0263
Epoch 32/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7230
Epoch 00032: val_loss improved from 0.02620 to 0.02616, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 514s 74ms/step - loss: 0.7230 -
val_loss: 0.0262
Epoch 33/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7226
```

```
Epoch 00033: val_loss did not improve from 0.02616

Epoch 00033: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
6910/6910 [==============================] - 486s 70ms/step - loss: 0.7226 -
val_loss: 0.0263
Epoch 34/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7206
Epoch 00034: val_loss improved from 0.02616 to 0.02441, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 496s 72ms/step - loss: 0.7206 -
val_loss: 0.0244
Epoch 35/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7193
Epoch 00035: val_loss improved from 0.02441 to 0.02397, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 472s 68ms/step - loss: 0.7193 -
val_loss: 0.0240
Epoch 36/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7191
Epoch 00036: val_loss improved from 0.02397 to 0.02379, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 468s 68ms/step - loss: 0.7191 -
val_loss: 0.0238
Epoch 37/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7188
Epoch 00037: val_loss improved from 0.02379 to 0.02357, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7188 -
val_loss: 0.0236
Epoch 38/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7187
Epoch 00038: val_loss improved from 0.02357 to 0.02348, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 478s 69ms/step - loss: 0.7187 -
val_loss: 0.0235
Epoch 39/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7183
Epoch 00039: val_loss improved from 0.02348 to 0.02332, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 460s 67ms/step - loss: 0.7183 -
val_loss: 0.0233
Epoch 40/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7179
Epoch 00040: val_loss did not improve from 0.02332
6910/6910 [==============================] - 468s 68ms/step - loss: 0.7179 -
val_loss: 0.0233
Epoch 41/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7183
```

```
Epoch 00041: val_loss improved from 0.02332 to 0.02320, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 461s 67ms/step - loss: 0.7183 -
val_loss: 0.0232
Epoch 42/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7180
Epoch 00042: val_loss did not improve from 0.02320
6910/6910 [==============================] - 466s 67ms/step - loss: 0.7180 -
val_loss: 0.0232
Epoch 43/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7178
Epoch 00043: val_loss improved from 0.02320 to 0.02312, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 455s 66ms/step - loss: 0.7178 -
val_loss: 0.0231
Epoch 44/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7180
Epoch 00044: val_loss improved from 0.02312 to 0.02305, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7180 -
val_loss: 0.0230
Epoch 45/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7179
Epoch 00045: val_loss improved from 0.02305 to 0.02298, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 467s 68ms/step - loss: 0.7179 -
val_loss: 0.0230
Epoch 46/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7173
Epoch 00046: val_loss improved from 0.02298 to 0.02295, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7173 -
val_loss: 0.0229
Epoch 47/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7176
Epoch 00047: val_loss improved from 0.02295 to 0.02288, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7176 -
val_loss: 0.0229
Epoch 48/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7171
Epoch 00048: val_loss improved from 0.02288 to 0.02276, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7171 -
val_loss: 0.0228
Epoch 49/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7173
Epoch 00049: val_loss did not improve from 0.02276
```

```
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7173 -
val_loss: 0.0228
Epoch 50/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7169
Epoch 00050: val_loss improved from 0.02276 to 0.02275, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7169 -
val_loss: 0.0227
Epoch 51/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7173
Epoch 00051: val_loss improved from 0.02275 to 0.02267, saving model to
Attention_concat_lstm_bigram.h5

Epoch 00051: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7173 -
val_loss: 0.0227
Epoch 52/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7171
Epoch 00052: val_loss improved from 0.02267 to 0.02262, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 466s 67ms/step - loss: 0.7171 -
val_loss: 0.0226
Epoch 53/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7170
Epoch 00053: val_loss improved from 0.02262 to 0.02258, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 458s 66ms/step - loss: 0.7170 -
val_loss: 0.0226
Epoch 54/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7168
Epoch 00054: val_loss improved from 0.02258 to 0.02256, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 458s 66ms/step - loss: 0.7168 -
val_loss: 0.0226
Epoch 55/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7165
Epoch 00055: val_loss improved from 0.02256 to 0.02251, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 457s 66ms/step - loss: 0.7165 -
val_loss: 0.0225
Epoch 56/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7169
Epoch 00056: val_loss improved from 0.02251 to 0.02251, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7169 -
val_loss: 0.0225
Epoch 57/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7173
```

```
Epoch 00057: val_loss improved from 0.02251 to 0.02251, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7173 -
val_loss: 0.0225
Epoch 58/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7169
Epoch 00058: val_loss improved from 0.02251 to 0.02250, saving model to
Attention_concat_lstm_bigram.h5

Epoch 00058: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
6910/6910 [==============================] - 455s 66ms/step - loss: 0.7169 -
val_loss: 0.0225
Epoch 59/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7168
Epoch 00059: val_loss did not improve from 0.02250
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7168 -
val_loss: 0.0225
Epoch 60/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7164
Epoch 00060: val_loss improved from 0.02250 to 0.02250, saving model to
Attention_concat_lstm_bigram.h5
6910/6910 [==============================] - 456s 66ms/step - loss: 0.7164 -
val_loss: 0.0225
Epoch 61/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7172
Epoch 00061: val_loss did not improve from 0.02250

Epoch 00061: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
6910/6910 [==============================] - 454s 66ms/step - loss: 0.7172 -
val_loss: 0.0225
Epoch 62/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7168
Epoch 00062: val_loss did not improve from 0.02250
6910/6910 [==============================] - 458s 66ms/step - loss: 0.7168 -
val_loss: 0.0225
Epoch 63/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7165
Epoch 00063: val_loss did not improve from 0.02250
6910/6910 [==============================] - 459s 66ms/step - loss: 0.7165 -
val_loss: 0.0225
Epoch 64/100
6910/6910 [==============================] - ETA: 0s - loss: 0.7171
Epoch 00064: val_loss did not improve from 0.02250

Epoch 00064: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
6910/6910 [==============================] - 454s 66ms/step - loss: 0.7171 -
val_loss: 0.0225
Epoch 65/100
```

```
6910/6910 [==============================] - ETA: 0s - loss: 0.7165
Epoch 00065: val_loss did not improve from 0.02250
6910/6910 [==============================] - 454s 66ms/step - loss: 0.7165 -
val_loss: 0.0225
Epoch 00065: early stopping
```

[85]:
```
<tensorflow.python.keras.callbacks.History at 0x224063cde88>
```

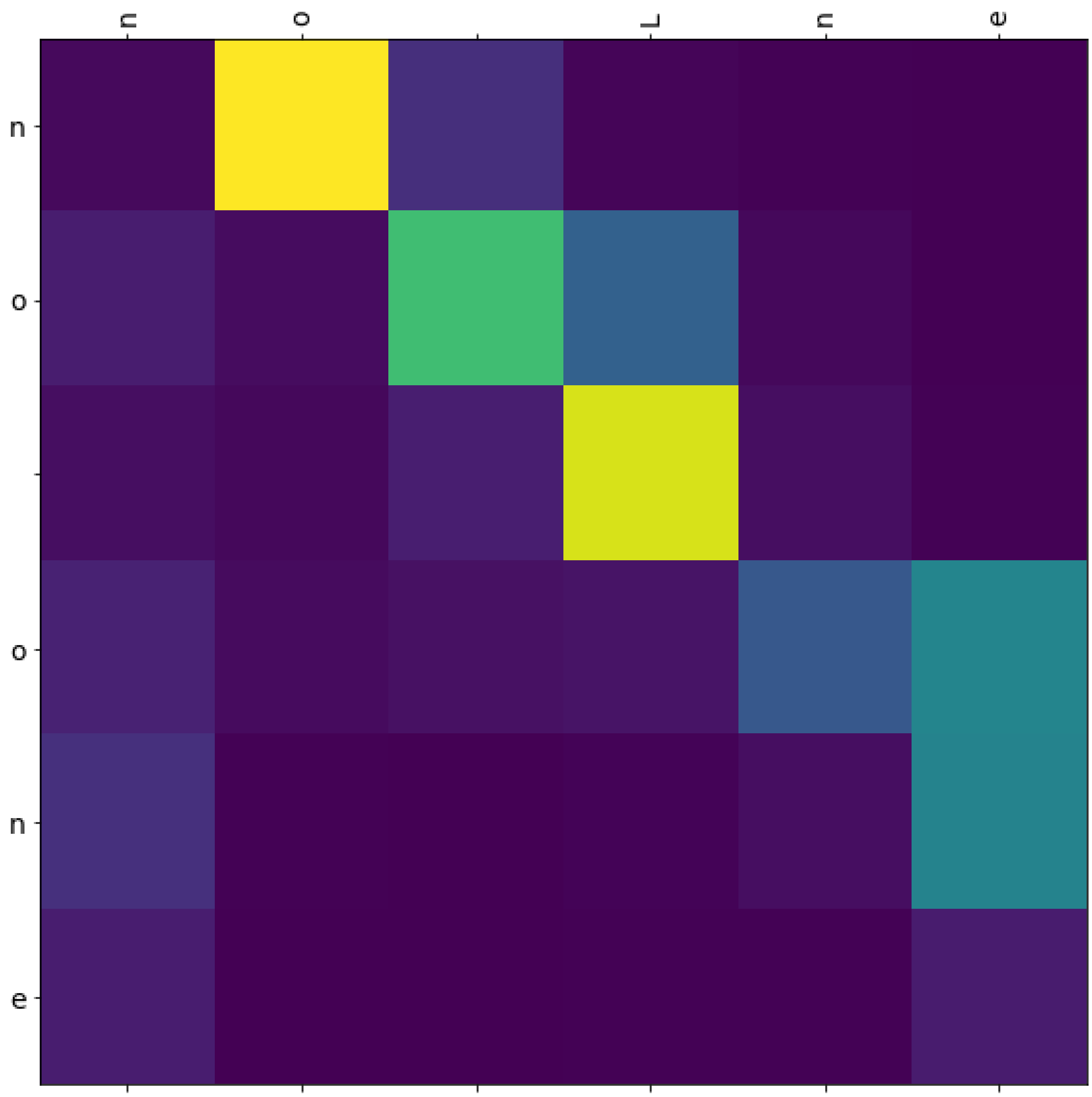
[87]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
↪lstm_size, lstm_size, maxlen, maxlen, 'concat', att_units,␣
↪bigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('Attention_concat_lstm_bigram.h5')
```

[88]:
```
sentence = bigram_train['input'].values[4]
result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
↪gram = 'bi')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[4])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  no Lne
predicted output :  no one
actual output : no one
```



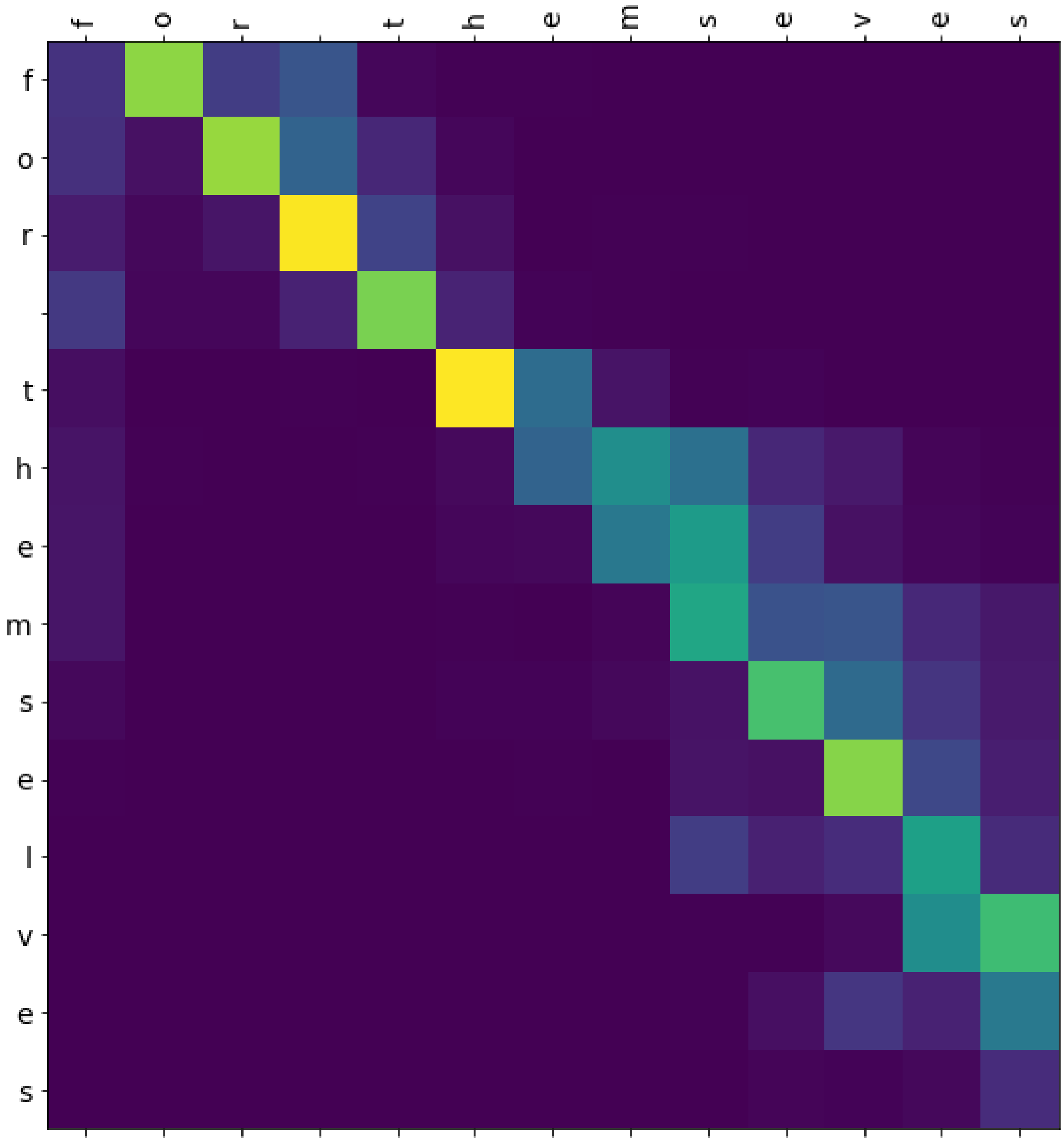
[89]:
```
sentence = bigram_train['input'].values[6]
result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
 ↪gram = 'bi')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[6])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  for themseves
predicted output :  for themselves
```

```
actual output : for themselves
```

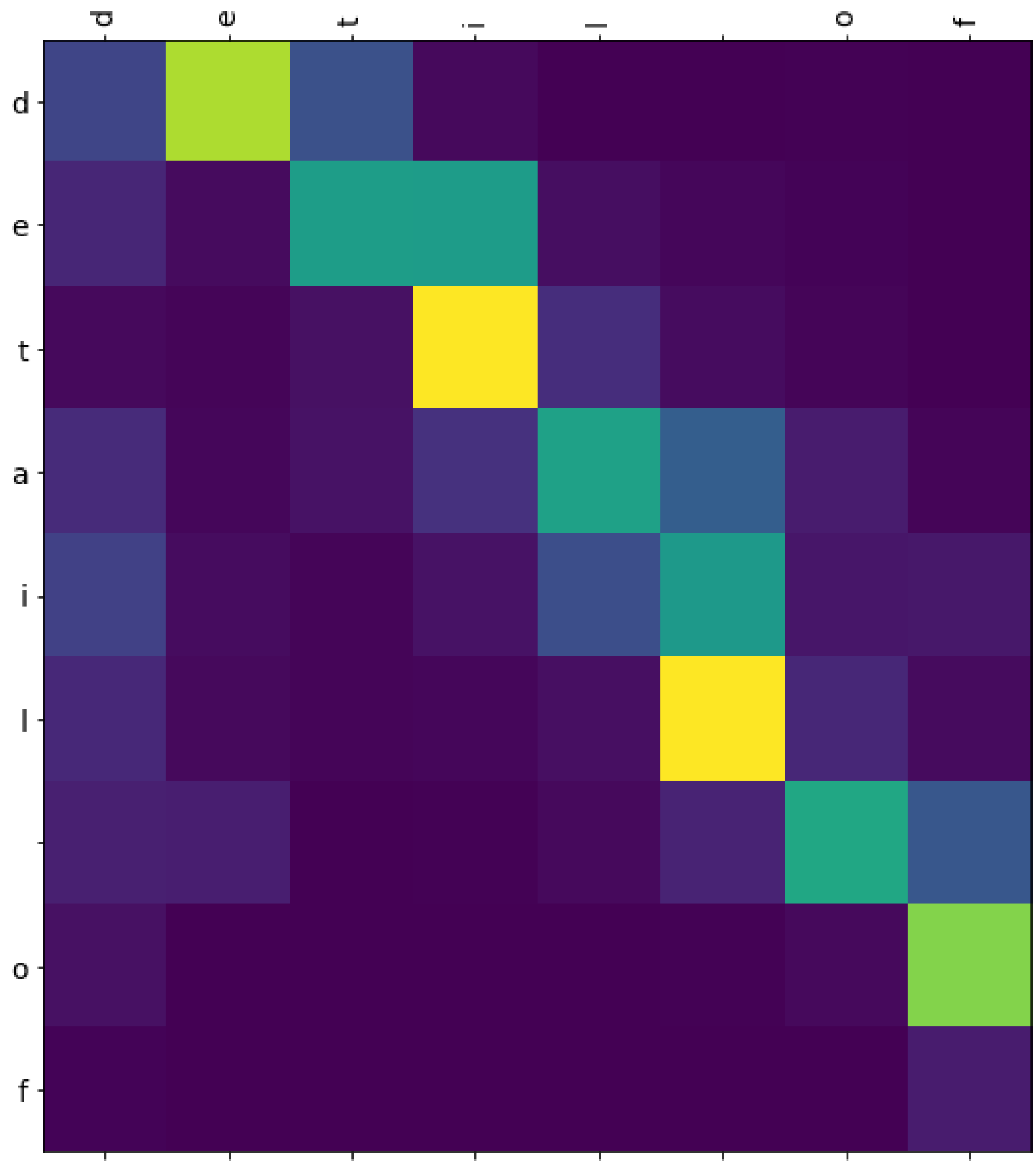



```
[90]: sentence = bigram_train['input'].values[7]
      result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
       ↪gram = 'bi')

      print('input : ', sentence)
      print('predicted output : ',result)
      print('actual output :', bigram_train['output'].values[7])
```

```
attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  detil of
predicted output :  detail of
actual output : detail of
```



[5]:
```
val_bleu = 0
for i in range(bigram_val.shape[0]):
```

```python
    inp = bigram_val['input'].values[i]
    out = bigram_val['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in range(bigram_train.shape[0]):
    inp = bigram_train['input'].values[i];
    out = bigram_train['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in range(bigram_test.shape[0]):
    inp = bigram_test['input'].values[i]
    out = bigram_test['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/bigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/bigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/bigram_test.shape[0])
```

```
BLEU Score on train:  0.9719945427791346
BLEU Score on val:  0.9588801560719675
BLEU Score on test:  0.9460340413418259
```

2.3 TriGram

[91]:
```python
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 34
```

[92]:
```python
model = encoder_decoder(vocab_size, vocab_size, embedding_dim, lstm_size,␣
 ↪lstm_size, maxlen, maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)

callbacks = [ModelCheckpoint('Attention_concat_lstm_trigram.h5',␣
 ↪save_best_only= True, verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = trigram_train_dataset,
          steps_per_epoch = trigram_train.shape[0]//batch_size,
          validation_data = trigram_val_dataset,
          validation_steps = trigram_val.shape[0]//batch_size,
          epochs = 100,
```

```
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/100
6910/6910 [==============================] - ETA: 0s - loss: 0.9333
Epoch 00001: val_loss improved from inf to 0.05902, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 619s 90ms/step - loss: 0.9333 -
val_loss: 0.0590
Epoch 2/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8491
Epoch 00002: val_loss improved from 0.05902 to 0.04622, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 622s 90ms/step - loss: 0.8491 -
val_loss: 0.0462
Epoch 3/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8422
Epoch 00003: val_loss improved from 0.04622 to 0.04152, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 625s 90ms/step - loss: 0.8422 -
val_loss: 0.0415
Epoch 4/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8382
Epoch 00004: val_loss improved from 0.04152 to 0.03638, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 632s 91ms/step - loss: 0.8382 -
val_loss: 0.0364
Epoch 5/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8360
Epoch 00005: val_loss improved from 0.03638 to 0.03372, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 629s 91ms/step - loss: 0.8360 -
val_loss: 0.0337
Epoch 6/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8345
Epoch 00006: val_loss improved from 0.03372 to 0.03284, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 626s 91ms/step - loss: 0.8345 -
val_loss: 0.0328
Epoch 7/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8329
Epoch 00007: val_loss improved from 0.03284 to 0.03111, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 626s 91ms/step - loss: 0.8329 -
val_loss: 0.0311
Epoch 8/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8318
Epoch 00008: val_loss improved from 0.03111 to 0.03011, saving model to
```

```
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 627s 91ms/step - loss: 0.8318 -
val_loss: 0.0301
Epoch 9/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8309
Epoch 00009: val_loss improved from 0.03011 to 0.02913, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 626s 91ms/step - loss: 0.8309 -
val_loss: 0.0291
Epoch 10/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8303
Epoch 00010: val_loss improved from 0.02913 to 0.02846, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 626s 91ms/step - loss: 0.8303 -
val_loss: 0.0285
Epoch 11/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8299
Epoch 00011: val_loss improved from 0.02846 to 0.02764, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 625s 90ms/step - loss: 0.8299 -
val_loss: 0.0276
Epoch 12/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8291
Epoch 00012: val_loss improved from 0.02764 to 0.02715, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 635s 92ms/step - loss: 0.8291 -
val_loss: 0.0272
Epoch 13/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8290
Epoch 00013: val_loss improved from 0.02715 to 0.02653, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 628s 91ms/step - loss: 0.8290 -
val_loss: 0.0265
Epoch 14/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8283
Epoch 00014: val_loss improved from 0.02653 to 0.02601, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 630s 91ms/step - loss: 0.8283 -
val_loss: 0.0260
Epoch 15/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8282
Epoch 00015: val_loss improved from 0.02601 to 0.02585, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 631s 91ms/step - loss: 0.8282 -
val_loss: 0.0258
Epoch 16/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8276
Epoch 00016: val_loss improved from 0.02585 to 0.02510, saving model to
```

```
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 632s 91ms/step - loss: 0.8276 -
val_loss: 0.0251
Epoch 17/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8272
Epoch 00017: val_loss improved from 0.02510 to 0.02482, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 632s 91ms/step - loss: 0.8272 -
val_loss: 0.0248
Epoch 18/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8267
Epoch 00018: val_loss improved from 0.02482 to 0.02455, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 631s 91ms/step - loss: 0.8267 -
val_loss: 0.0245
Epoch 19/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8264
Epoch 00019: val_loss improved from 0.02455 to 0.02399, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 635s 92ms/step - loss: 0.8264 -
val_loss: 0.0240
Epoch 20/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8261
Epoch 00020: val_loss did not improve from 0.02399
6910/6910 [==============================] - 634s 92ms/step - loss: 0.8261 -
val_loss: 0.0240
Epoch 21/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8261
Epoch 00021: val_loss did not improve from 0.02399
6910/6910 [==============================] - 640s 93ms/step - loss: 0.8261 -
val_loss: 0.0243
Epoch 22/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8257
Epoch 00022: val_loss improved from 0.02399 to 0.02359, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 632s 92ms/step - loss: 0.8257 -
val_loss: 0.0236
Epoch 23/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8259
Epoch 00023: val_loss improved from 0.02359 to 0.02338, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 633s 92ms/step - loss: 0.8259 -
val_loss: 0.0234
Epoch 24/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8253
Epoch 00024: val_loss did not improve from 0.02338
6910/6910 [==============================] - 631s 91ms/step - loss: 0.8253 -
val_loss: 0.0234
```

```
Epoch 25/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8251
Epoch 00025: val_loss improved from 0.02338 to 0.02304, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 647s 94ms/step - loss: 0.8251 -
val_loss: 0.0230
Epoch 26/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8251
Epoch 00026: val_loss improved from 0.02304 to 0.02287, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 657s 95ms/step - loss: 0.8251 -
val_loss: 0.0229
Epoch 27/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8246
Epoch 00027: val_loss improved from 0.02287 to 0.02246, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 701s 101ms/step - loss: 0.8246 -
val_loss: 0.0225
Epoch 28/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8247
Epoch 00028: val_loss improved from 0.02246 to 0.02219, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 700s 101ms/step - loss: 0.8247 -
val_loss: 0.0222
Epoch 29/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8247
Epoch 00029: val_loss did not improve from 0.02219
6910/6910 [==============================] - 657s 95ms/step - loss: 0.8247 -
val_loss: 0.0226
Epoch 30/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8284
Epoch 00030: val_loss did not improve from 0.02219
6910/6910 [==============================] - 648s 94ms/step - loss: 0.8284 -
val_loss: 0.0229
Epoch 31/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8247
Epoch 00031: val_loss did not improve from 0.02219

Epoch 00031: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
6910/6910 [==============================] - 645s 93ms/step - loss: 0.8247 -
val_loss: 0.0227
Epoch 32/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8227
Epoch 00032: val_loss improved from 0.02219 to 0.02104, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 660s 96ms/step - loss: 0.8227 -
val_loss: 0.0210
Epoch 33/100
```

```
6910/6910 [==============================] - ETA: 0s - loss: 0.8221
Epoch 00033: val_loss improved from 0.02104 to 0.02061, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 679s 98ms/step - loss: 0.8221 -
val_loss: 0.0206
Epoch 34/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8214
Epoch 00034: val_loss improved from 0.02061 to 0.02026, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 656s 95ms/step - loss: 0.8214 -
val_loss: 0.0203
Epoch 35/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8212
Epoch 00035: val_loss did not improve from 0.02026
6910/6910 [==============================] - 658s 95ms/step - loss: 0.8212 -
val_loss: 0.0203
Epoch 36/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8211
Epoch 00036: val_loss improved from 0.02026 to 0.02003, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 648s 94ms/step - loss: 0.8211 -
val_loss: 0.0200
Epoch 37/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8209
Epoch 00037: val_loss improved from 0.02003 to 0.01989, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 656s 95ms/step - loss: 0.8209 -
val_loss: 0.0199
Epoch 38/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8207
Epoch 00038: val_loss improved from 0.01989 to 0.01973, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 651s 94ms/step - loss: 0.8207 -
val_loss: 0.0197
Epoch 39/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8205
Epoch 00039: val_loss did not improve from 0.01973
6910/6910 [==============================] - 641s 93ms/step - loss: 0.8205 -
val_loss: 0.0197
Epoch 40/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8203
Epoch 00040: val_loss improved from 0.01973 to 0.01965, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 694s 100ms/step - loss: 0.8203 -
val_loss: 0.0196
Epoch 41/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8205
Epoch 00041: val_loss improved from 0.01965 to 0.01948, saving model to
```

```
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 649s 94ms/step - loss: 0.8205 -
val_loss: 0.0195
Epoch 42/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8204
Epoch 00042: val_loss improved from 0.01948 to 0.01943, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 653s 94ms/step - loss: 0.8204 -
val_loss: 0.0194
Epoch 43/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8200
Epoch 00043: val_loss improved from 0.01943 to 0.01934, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 671s 97ms/step - loss: 0.8200 -
val_loss: 0.0193
Epoch 44/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8200
Epoch 00044: val_loss did not improve from 0.01934
6910/6910 [==============================] - 679s 98ms/step - loss: 0.8200 -
val_loss: 0.0194
Epoch 45/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8203
Epoch 00045: val_loss improved from 0.01934 to 0.01928, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 658s 95ms/step - loss: 0.8203 -
val_loss: 0.0193
Epoch 46/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8198- ETA:
Epoch 00046: val_loss improved from 0.01928 to 0.01921, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 653s 95ms/step - loss: 0.8198 -
val_loss: 0.0192
Epoch 47/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8194
Epoch 00047: val_loss improved from 0.01921 to 0.01915, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 650s 94ms/step - loss: 0.8194 -
val_loss: 0.0191
Epoch 48/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8199
Epoch 00048: val_loss did not improve from 0.01915
6910/6910 [==============================] - 661s 96ms/step - loss: 0.8199 -
val_loss: 0.0192
Epoch 49/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8197
Epoch 00049: val_loss improved from 0.01915 to 0.01910, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 659s 95ms/step - loss: 0.8197 -
```

```
val_loss: 0.0191
Epoch 50/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8198
Epoch 00050: val_loss improved from 0.01910 to 0.01904, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 642s 93ms/step - loss: 0.8198 -
val_loss: 0.0190
Epoch 51/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8194
Epoch 00051: val_loss improved from 0.01904 to 0.01896, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 646s 93ms/step - loss: 0.8194 -
val_loss: 0.0190
Epoch 52/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8194
Epoch 00052: val_loss improved from 0.01896 to 0.01895, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 652s 94ms/step - loss: 0.8194 -
val_loss: 0.0190
Epoch 53/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8198
Epoch 00053: val_loss improved from 0.01895 to 0.01893, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 670s 97ms/step - loss: 0.8198 -
val_loss: 0.0189
Epoch 54/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8194
Epoch 00054: val_loss improved from 0.01893 to 0.01883, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 657s 95ms/step - loss: 0.8194 -
val_loss: 0.0188
Epoch 55/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8196
Epoch 00055: val_loss improved from 0.01883 to 0.01872, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 654s 95ms/step - loss: 0.8196 -
val_loss: 0.0187
Epoch 56/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8190
Epoch 00056: val_loss did not improve from 0.01872
6910/6910 [==============================] - 651s 94ms/step - loss: 0.8190 -
val_loss: 0.0188
Epoch 57/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8196
Epoch 00057: val_loss did not improve from 0.01872
6910/6910 [==============================] - 652s 94ms/step - loss: 0.8196 -
val_loss: 0.0188
Epoch 58/100
```

```
6910/6910 [==============================] - ETA: 0s - loss: 0.8190
Epoch 00058: val_loss did not improve from 0.01872

Epoch 00058: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 658s 95ms/step - loss: 0.8190 -
val_loss: 0.0187
Epoch 59/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8188
Epoch 00059: val_loss improved from 0.01872 to 0.01862, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 660s 95ms/step - loss: 0.8188 -
val_loss: 0.0186
Epoch 60/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8192
Epoch 00060: val_loss improved from 0.01862 to 0.01860, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 672s 97ms/step - loss: 0.8192 -
val_loss: 0.0186
Epoch 61/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8188
Epoch 00061: val_loss improved from 0.01860 to 0.01858, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 673s 97ms/step - loss: 0.8188 -
val_loss: 0.0186
Epoch 62/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8192
Epoch 00062: val_loss improved from 0.01858 to 0.01857, saving model to
Attention_concat_lstm_trigram.h5

Epoch 00062: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
6910/6910 [==============================] - 684s 99ms/step - loss: 0.8192 -
val_loss: 0.0186
Epoch 63/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8191
Epoch 00063: val_loss improved from 0.01857 to 0.01856, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 650s 94ms/step - loss: 0.8191 -
val_loss: 0.0186
Epoch 64/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8191
Epoch 00064: val_loss improved from 0.01856 to 0.01856, saving model to
Attention_concat_lstm_trigram.h5
6910/6910 [==============================] - 640s 93ms/step - loss: 0.8191 -
val_loss: 0.0186
Epoch 65/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8192
Epoch 00065: val_loss improved from 0.01856 to 0.01854, saving model to
Attention_concat_lstm_trigram.h5
```

```
Epoch 00065: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
6910/6910 [==============================] - 638s 92ms/step - loss: 0.8192 -
val_loss: 0.0185
Epoch 66/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8191
Epoch 00066: val_loss did not improve from 0.01854
6910/6910 [==============================] - 2044s 296ms/step - loss: 0.8191 -
val_loss: 0.0185
Epoch 67/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8187
Epoch 00067: val_loss did not improve from 0.01854
6910/6910 [==============================] - 642s 93ms/step - loss: 0.8187 -
val_loss: 0.0185
Epoch 68/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8187
Epoch 00068: val_loss did not improve from 0.01854

Epoch 00068: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
6910/6910 [==============================] - 672s 97ms/step - loss: 0.8187 -
val_loss: 0.0185
Epoch 69/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8188
Epoch 00069: val_loss did not improve from 0.01854
6910/6910 [==============================] - 689s 100ms/step - loss: 0.8188 -
val_loss: 0.0185
Epoch 70/100
6910/6910 [==============================] - ETA: 0s - loss: 0.8191
Epoch 00070: val_loss did not improve from 0.01854
6910/6910 [==============================] - 652s 94ms/step - loss: 0.8191 -
val_loss: 0.0185
Epoch 00070: early stopping
```

[92]:
```
<tensorflow.python.keras.callbacks.History at 0x2269658dcc8>
```

[93]:
```python
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, lstm_size, maxlen, maxlen, 'concat', att_units,␣
 ↪trigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('Attention_concat_lstm_trigram.h5')
```

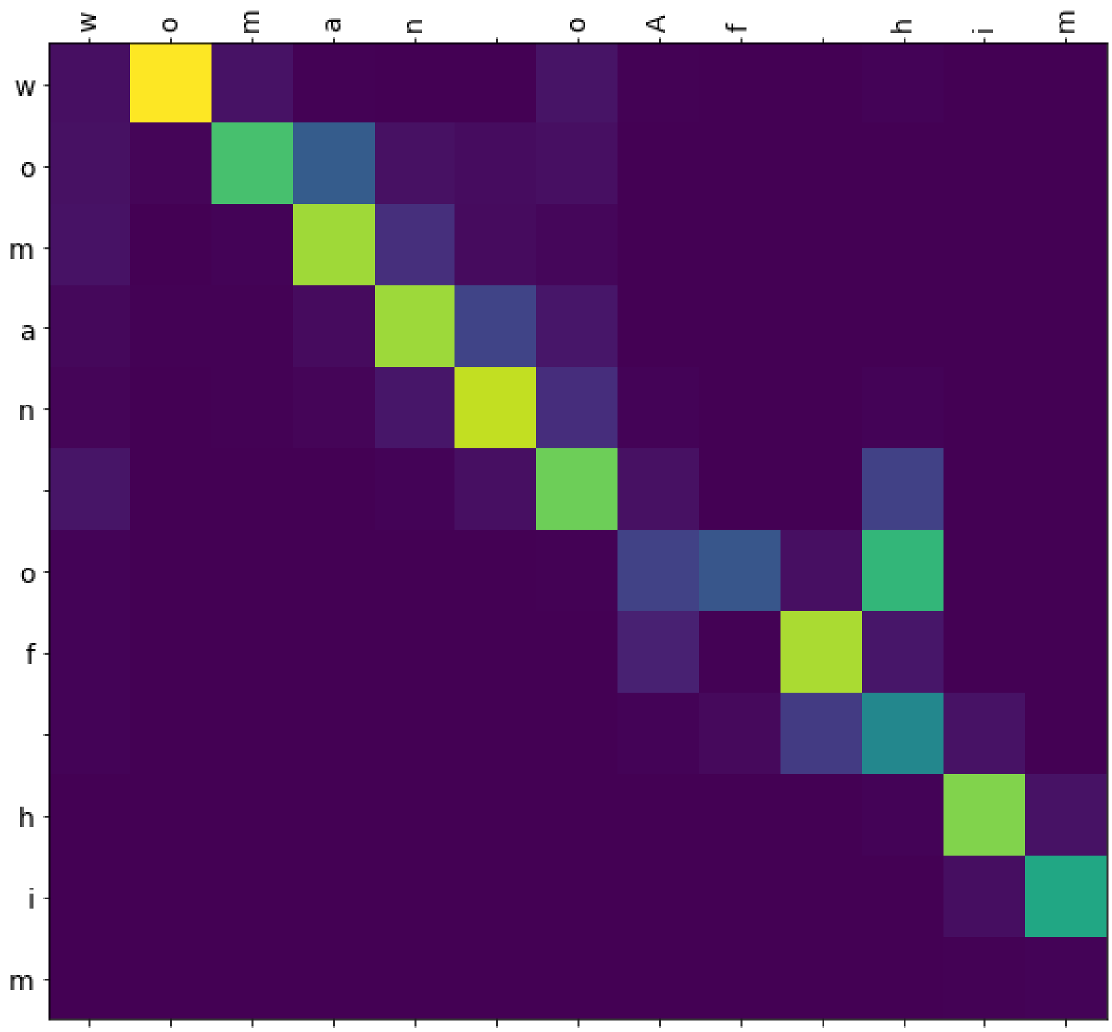
[97]:
```python
sentence = trigram_train['input'].values[4]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')

print('input : ', sentence)
```

```
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[4])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  woman oAf him
predicted output :  woman of him
actual output : woman of him
```



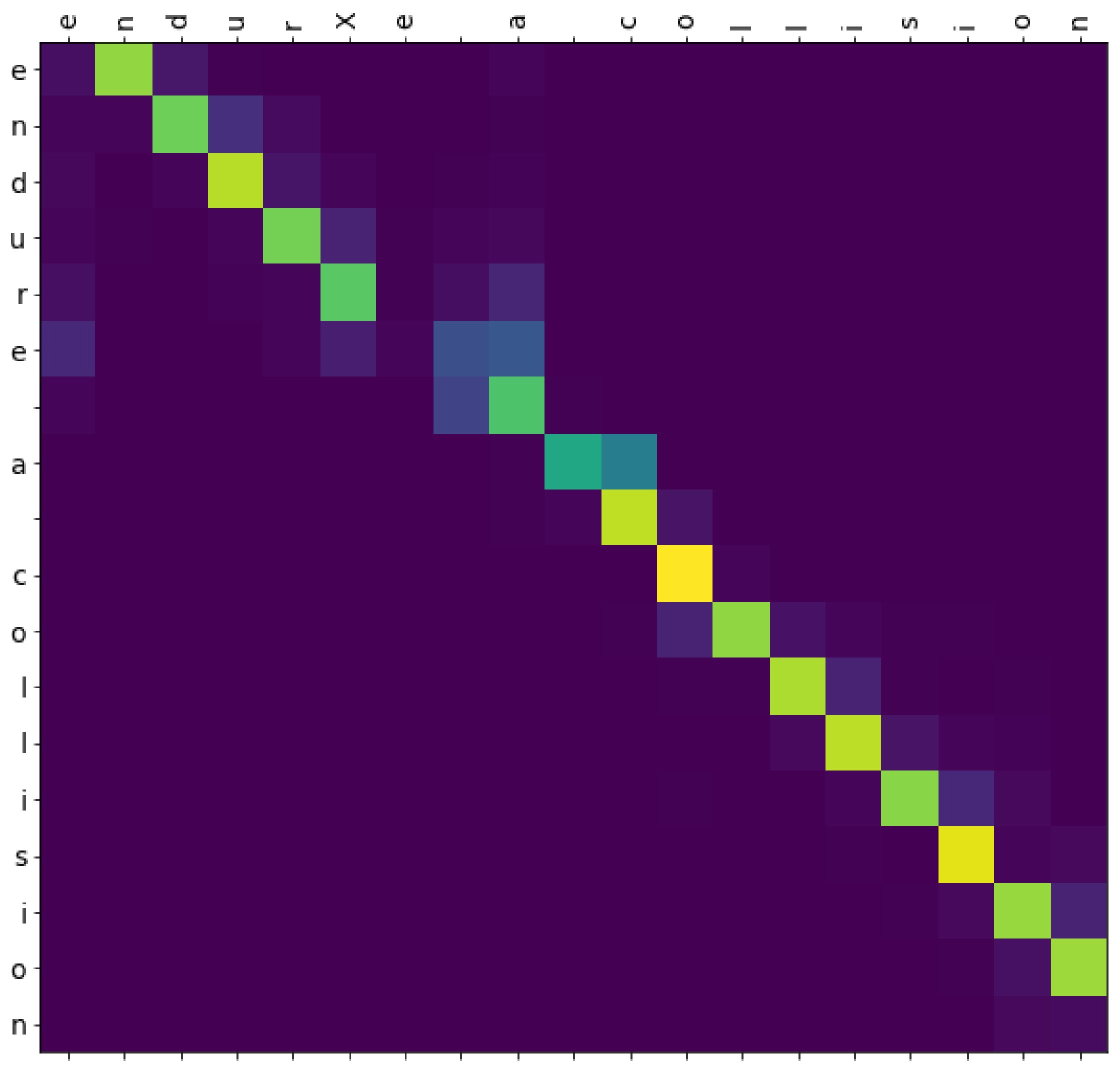
[98]:
```
sentence = trigram_train['input'].values[7]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')

print('input : ', sentence)
```

```
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[7])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  endurXe a collision
predicted output :  endure a collision
actual output : endure a collision
```

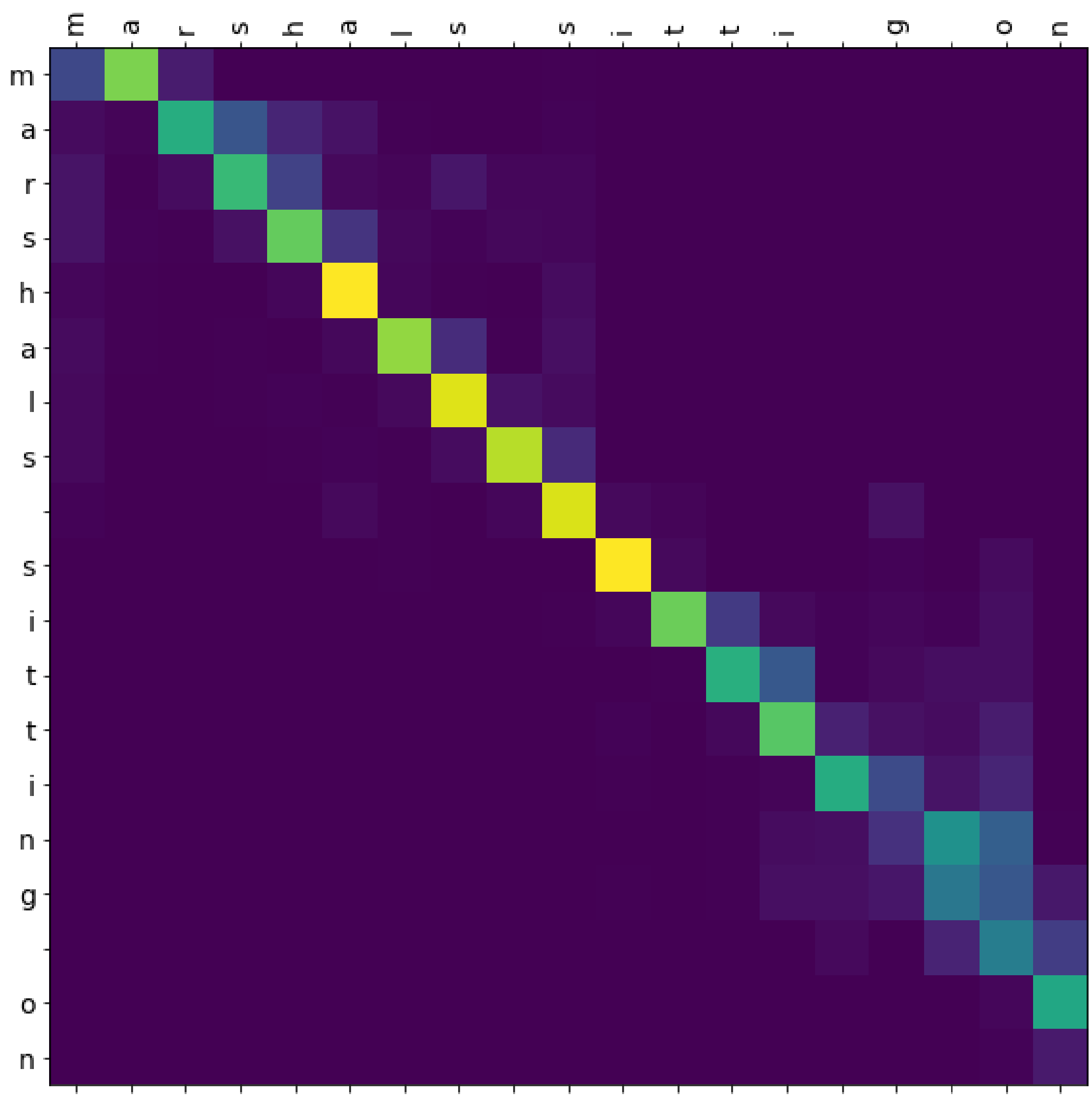


[99]:
```
sentence = trigram_train['input'].values[10]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')
```

```
print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[10])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  marshals sitti g on
predicted output :  marshals sitting on
actual output : marshals sitting on
```



[6]:
```
val_bleu = 0
for i in range(trigram_val.shape[0]):
```

```
    inp = trigram_val['input'].values[i]
    out = trigram_val['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in range(trigram_train.shape[0]):
    inp = trigram_train['input'].values[i];
    out = trigram_train['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in range(trigram_test.shape[0]):
    inp = trigram_test['input'].values[i]
    out = trigram_test['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/trigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/trigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/trigram_test.shape[0])
```

```
BLEU Score on train:  0.9811416412867453
BLEU Score on val:  0.9743928120876085
BLEU Score on test:  0.96227561809527
```

3. Seq2Seq Bi-LSTM with Attention Mechanism

[106]:
```
class Encoder(tf.keras.layers.Layer):
    def __init__(self, vocab_size, embedding_size, lstm_size, input_length):
        super(Encoder, self).__init__()
        self.lstm_size = lstm_size
        self.enc_embed = Embedding(input_dim = vocab_size, output_dim =␣
 ↪embedding_size)
        self.enc_lstm = Bidirectional(LSTM(lstm_size, return_sequences = True,␣
 ↪return_state = True, dropout = 0.4))

    def call(self, input_sequence, states):
        embedding = self.enc_embed(input_sequence)
        output_state, enc_frwd_h, enc_frwd_c, enc_bkwd_h, enc_bkwd_c = self.
 ↪enc_lstm(embedding, initial_state = states)
        return output_state, enc_frwd_h, enc_frwd_c, enc_bkwd_h, enc_bkwd_c

    def initialize_states(self, batch_size):
        return [tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size,␣
 ↪self.lstm_size)),
```

```
                tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size,␣
↪self.lstm_size))]

class Attention(tf.keras.layers.Layer):
    def __init__(self,scoring_function, att_units):
        super(Attention, self).__init__()
        self.scoring_function = scoring_function
        if scoring_function == 'dot':
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'general':
            self.W = Dense(att_units)
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'concat':
            self.W1 = Dense(att_units)
            self.W2 = Dense(att_units)
            self.W3 = Dense(att_units)
            self.V = Dense(1)

    def call(self, dec_frwd_state, dec_bkwd_state, encoder_output):
        dec_frwd_state = tf.expand_dims(dec_frwd_state, 1)
        dec_bkwd_state = tf.expand_dims(dec_bkwd_state, 1)
#
        if self.scoring_function == 'dot':
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state,␣
↪(0, 2, 1)), encoder_output]), (0, 2,1))
        elif self.scoring_function == 'general':
            mul = self.W(encoder_output)
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state,␣
↪(0, 2, 1)), mul]), (0, 2,1))
        elif self.scoring_function == 'concat':
            inter = self.W1(dec_frwd_state) + self.W2(dec_bkwd_state) + self.
↪W3(encoder_output)
            tan = tf.nn.tanh(inter)
            score = self.V(tan)
        attention_weights = tf.nn.softmax(score, axis =1)
        context_vector = attention_weights * encoder_output
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights

class OneStepDecoder(tf.keras.layers.Layer):
    def __init__(self, vocab_size, embedding_dim, input_length, dec_units␣
↪,score_fun ,att_units):
        super(OneStepDecoder, self).__init__()
      # Initialize decoder embedding layer, LSTM and any other objects needed
        self.embed_dec = Embedding(input_dim = vocab_size, output_dim =␣
↪embedding_dim)
```

```
        self.lstm = Bidirectional(LSTM(dec_units, return_sequences = True,␣
↪return_state = True, dropout = 0.4))
        self.attention = Attention(scoring_function = score_fun, att_units =␣
↪att_units)
        self.fc = Dense(vocab_size)

    def call(self,input_to_decoder, encoder_output, state_frwd_h, state_frwd_c,␣
↪state_bkwd_h, state_bkwd_c):
        embed = self.embed_dec(input_to_decoder)
        context_vect, attention_weights = self.attention(state_frwd_h,␣
↪state_bkwd_h, encoder_output)
        final_inp = tf.concat([tf.expand_dims(context_vect, 1), embed], axis =␣
↪-1)
        out, dec_frwd_h, dec_frwd_c, dec_bkwd_h, dec_bkwd_c = self.
↪lstm(final_inp, [state_frwd_h, state_frwd_c, state_bkwd_h, state_bkwd_c])
        out = tf.reshape(out, (-1, out.shape[2]))
        out = Dropout(0.5)(out)
        output = self.fc(out)
        return output, dec_frwd_h, dec_frwd_c, dec_bkwd_h, dec_bkwd_c,␣
↪attention_weights, context_vect

class encoder_decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, dec_units,␣
↪max_len, score_fun, att_units, batch_size):
        #Intialize objects from encoder decoder
        super(encoder_decoder, self).__init__()
        self.encoder = Encoder(vocab_size, embedding_dim, enc_units, max_len)
        self.one_step_decoder = OneStepDecoder(vocab_size, embedding_dim,␣
↪max_len, dec_units ,score_fun ,att_units)
        self.batch_size = batch_size

    def call(self, data):
        enc_inp, dec_inp = data[0], data[1]
        initial_state = self.encoder.initialize_states(self.batch_size)
        enc_output, enc_frwd_h, enc_frwd_c, enc_bkwd_h, enc_bkwd_c = self.
↪encoder(enc_inp, initial_state)
        all_outputs = tf.TensorArray(dtype = tf.float32, size= max_len)

        dec_frwd_h = enc_frwd_h
        dec_frwd_c = enc_frwd_c
        dec_bkwd_h = enc_bkwd_h
        dec_bkwd_c = enc_bkwd_c
        for timestep in range(max_len):
            # Call onestepdecoder for each token in decoder_input
```

```
            output, dec_frwd_h, dec_frwd_c, dec_bkwd_h, dec_bkwd_c, _, _ = self.
 ↪one_step_decoder(dec_inp[:, timestep:timestep+1], enc_output, dec_frwd_h,␣
 ↪dec_frwd_c, dec_bkwd_h, dec_bkwd_c)
            # Store the output in tensorarray
            all_outputs = all_outputs.write(timestep, output)
        # Return the tensor array
        all_outputs = tf.transpose(all_outputs.stack(), (1, 0, 2))
        # return the decoder output
        return all_outputs

class pred_Encoder_decoder(tf.keras.Model):
    def __init__(self, inp_vocab_size, out_vocab_size, embedding_dim,␣
 ↪enc_units, dec_units, max_len_ita, max_len_eng, score_fun, att_units,␣
 ↪word_to_index):
        #Intialize objects from encoder decoder
        super(pred_Encoder_decoder, self).__init__()
        self.encoder = Encoder(inp_vocab_size, embedding_dim, enc_units,␣
 ↪max_len_ita)
        self.one_step_decoder = OneStepDecoder(out_vocab_size, embedding_dim,␣
 ↪max_len_eng, dec_units, score_fun, att_units)
        self.word_to_index = word_to_index

    def call(self, params):
        enc_inp = params[0]
        initial_state = self.encoder.initialize_states(1)
        enc_output, enc_frwd_h, enc_frwd_c, enc_bkwd_h, enc_bkwd_c = self.
 ↪encoder(enc_inp, initial_state)
        pred = tf.expand_dims([self.word_to_index['<SOW>']], 0)
        all_pred = []
        all_attention = []

        dec_frwd_h = enc_frwd_h
        dec_frwd_c = enc_frwd_c
        dec_bkwd_h = enc_bkwd_h
        dec_bkwd_c = enc_bkwd_c
        for timestep in range(max_len):
            # Call onestepdecoder for each token in decoder_input
            output, dec_frwd_h, dec_frwd_c, dec_bkwd_h, dec_bkwd_c, attention,␣
 ↪_ = self.one_step_decoder(pred, enc_output, dec_frwd_h, dec_frwd_c,␣
 ↪dec_bkwd_h, dec_bkwd_c)
            pred = tf.argmax(output, axis = -1)
            all_pred.append(pred)
            pred = tf.expand_dims(pred, 0)
            all_attention.append(attention)

        return all_pred, all_attention
```

3.1 UniGram

[ ]:
```
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 22
```

[41]:
```
model = encoder_decoder(vocab_size, embedding_dim, lstm_size, lstm_size,␣
 ↪maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)

callbacks = [ModelCheckpoint('concat_best.h5', save_best_only= True, verbose =␣
 ↪1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = unigram_train_dataset,
          steps_per_epoch = unigram_train.shape[0]//batch_size,
          validation_data = unigram_val_dataset,
          validation_steps = unigram_val.shape[0]//batch_size,
          epochs = 50,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/50
258/258 [==============================] - ETA: 0s - loss: 0.9726
Epoch 00001: val_loss improved from inf to 0.53295, saving model to
concat_best.h5
258/258 [==============================] - 33s 127ms/step - loss: 0.9726 -
val_loss: 0.5329
Epoch 2/50
258/258 [==============================] - ETA: 0s - loss: 0.3148
Epoch 00002: val_loss improved from 0.53295 to 0.16296, saving model to
concat_best.h5
258/258 [==============================] - 23s 89ms/step - loss: 0.3148 -
val_loss: 0.1630
Epoch 3/50
258/258 [==============================] - ETA: 0s - loss: 0.1810
Epoch 00003: val_loss improved from 0.16296 to 0.13154, saving model to
concat_best.h5
258/258 [==============================] - 24s 91ms/step - loss: 0.1810 -
val_loss: 0.1315
Epoch 4/50
258/258 [==============================] - ETA: 0s - loss: 0.1521
Epoch 00004: val_loss improved from 0.13154 to 0.12262, saving model to
concat_best.h5
258/258 [==============================] - 24s 93ms/step - loss: 0.1521 -
val_loss: 0.1226
```

```
Epoch 5/50
258/258 [==============================] - ETA: 0s - loss: 0.1372
Epoch 00005: val_loss improved from 0.12262 to 0.11498, saving model to
concat_best.h5
258/258 [==============================] - 24s 93ms/step - loss: 0.1372 -
val_loss: 0.1150
Epoch 6/50
258/258 [==============================] - ETA: 0s - loss: 0.1274
Epoch 00006: val_loss improved from 0.11498 to 0.11055, saving model to
concat_best.h5
258/258 [==============================] - 24s 93ms/step - loss: 0.1274 -
val_loss: 0.1105
Epoch 7/50
258/258 [==============================] - ETA: 0s - loss: 0.1208
Epoch 00007: val_loss improved from 0.11055 to 0.10576, saving model to
concat_best.h5
258/258 [==============================] - 24s 92ms/step - loss: 0.1208 -
val_loss: 0.1058
Epoch 8/50
258/258 [==============================] - ETA: 0s - loss: 0.1139
Epoch 00008: val_loss did not improve from 0.10576
258/258 [==============================] - 24s 93ms/step - loss: 0.1139 -
val_loss: 0.1067
Epoch 9/50
258/258 [==============================] - ETA: 0s - loss: 0.1096
Epoch 00009: val_loss improved from 0.10576 to 0.10549, saving model to
concat_best.h5
258/258 [==============================] - 24s 93ms/step - loss: 0.1096 -
val_loss: 0.1055
Epoch 10/50
258/258 [==============================] - ETA: 0s - loss: 0.1053
Epoch 00010: val_loss improved from 0.10549 to 0.10507, saving model to
concat_best.h5
258/258 [==============================] - 24s 94ms/step - loss: 0.1053 -
val_loss: 0.1051
Epoch 11/50
258/258 [==============================] - ETA: 0s - loss: 0.1010
Epoch 00011: val_loss improved from 0.10507 to 0.10283, saving model to
concat_best.h5
258/258 [==============================] - 24s 94ms/step - loss: 0.1010 -
val_loss: 0.1028
Epoch 12/50
258/258 [==============================] - ETA: 0s - loss: 0.0977
Epoch 00012: val_loss improved from 0.10283 to 0.10107, saving model to
concat_best.h5
258/258 [==============================] - 24s 95ms/step - loss: 0.0977 -
val_loss: 0.1011
Epoch 13/50
```

```
258/258 [==============================] - ETA: 0s - loss: 0.0922
Epoch 00013: val_loss improved from 0.10107 to 0.10079, saving model to
concat_best.h5
258/258 [==============================] - 25s 95ms/step - loss: 0.0922 -
val_loss: 0.1008
Epoch 14/50
258/258 [==============================] - ETA: 0s - loss: 0.0890
Epoch 00014: val_loss did not improve from 0.10079
258/258 [==============================] - 25s 95ms/step - loss: 0.0890 -
val_loss: 0.1013
Epoch 15/50
258/258 [==============================] - ETA: 0s - loss: 0.0858
Epoch 00015: val_loss did not improve from 0.10079
258/258 [==============================] - 25s 95ms/step - loss: 0.0858 -
val_loss: 0.1018
Epoch 16/50
258/258 [==============================] - ETA: 0s - loss: 0.0827- ETA
Epoch 00016: val_loss did not improve from 0.10079

Epoch 00016: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
258/258 [==============================] - 25s 96ms/step - loss: 0.0827 -
val_loss: 0.1014
Epoch 17/50
258/258 [==============================] - ETA: 0s - loss: 0.0728
Epoch 00017: val_loss improved from 0.10079 to 0.09775, saving model to
concat_best.h5
258/258 [==============================] - 25s 96ms/step - loss: 0.0728 -
val_loss: 0.0977
Epoch 18/50
258/258 [==============================] - ETA: 0s - loss: 0.0703
Epoch 00018: val_loss improved from 0.09775 to 0.09762, saving model to
concat_best.h5
258/258 [==============================] - 25s 96ms/step - loss: 0.0703 -
val_loss: 0.0976
Epoch 19/50
258/258 [==============================] - ETA: 0s - loss: 0.0689
Epoch 00019: val_loss improved from 0.09762 to 0.09739, saving model to
concat_best.h5
258/258 [==============================] - 25s 96ms/step - loss: 0.0689 -
val_loss: 0.0974
Epoch 20/50
258/258 [==============================] - ETA: 0s - loss: 0.0680
Epoch 00020: val_loss did not improve from 0.09739
258/258 [==============================] - 25s 96ms/step - loss: 0.0680 -
val_loss: 0.0975
Epoch 21/50
258/258 [==============================] - ETA: 0s - loss: 0.0665
Epoch 00021: val_loss improved from 0.09739 to 0.09734, saving model to
```

```
concat_best.h5
258/258 [==============================] - 25s 96ms/step - loss: 0.0665 -
val_loss: 0.0973
Epoch 22/50
258/258 [==============================] - ETA: 0s - loss: 0.0660
Epoch 00022: val_loss did not improve from 0.09734

Epoch 00022: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
258/258 [==============================] - 25s 96ms/step - loss: 0.0660 -
val_loss: 0.0978
Epoch 23/50
258/258 [==============================] - ETA: 0s - loss: 0.0643
Epoch 00023: val_loss did not improve from 0.09734
258/258 [==============================] - 25s 96ms/step - loss: 0.0643 -
val_loss: 0.0977
Epoch 24/50
258/258 [==============================] - ETA: 0s - loss: 0.0644
Epoch 00024: val_loss did not improve from 0.09734
258/258 [==============================] - 25s 96ms/step - loss: 0.0644 -
val_loss: 0.0976
Epoch 25/50
258/258 [==============================] - ETA: 0s - loss: 0.0641
Epoch 00025: val_loss did not improve from 0.09734

Epoch 00025: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
258/258 [==============================] - 25s 96ms/step - loss: 0.0641 -
val_loss: 0.0977
Epoch 26/50
258/258 [==============================] - ETA: 0s - loss: 0.0638
Epoch 00026: val_loss did not improve from 0.09734
258/258 [==============================] - 25s 96ms/step - loss: 0.0638 -
val_loss: 0.0976
Epoch 00026: early stopping
```

[41]:
```
<tensorflow.python.keras.callbacks.History at 0x29a98dd50c8>
```

[43]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, lstm_size, max_len, max_len, 'concat', att_units)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('concat_best.h5')
```

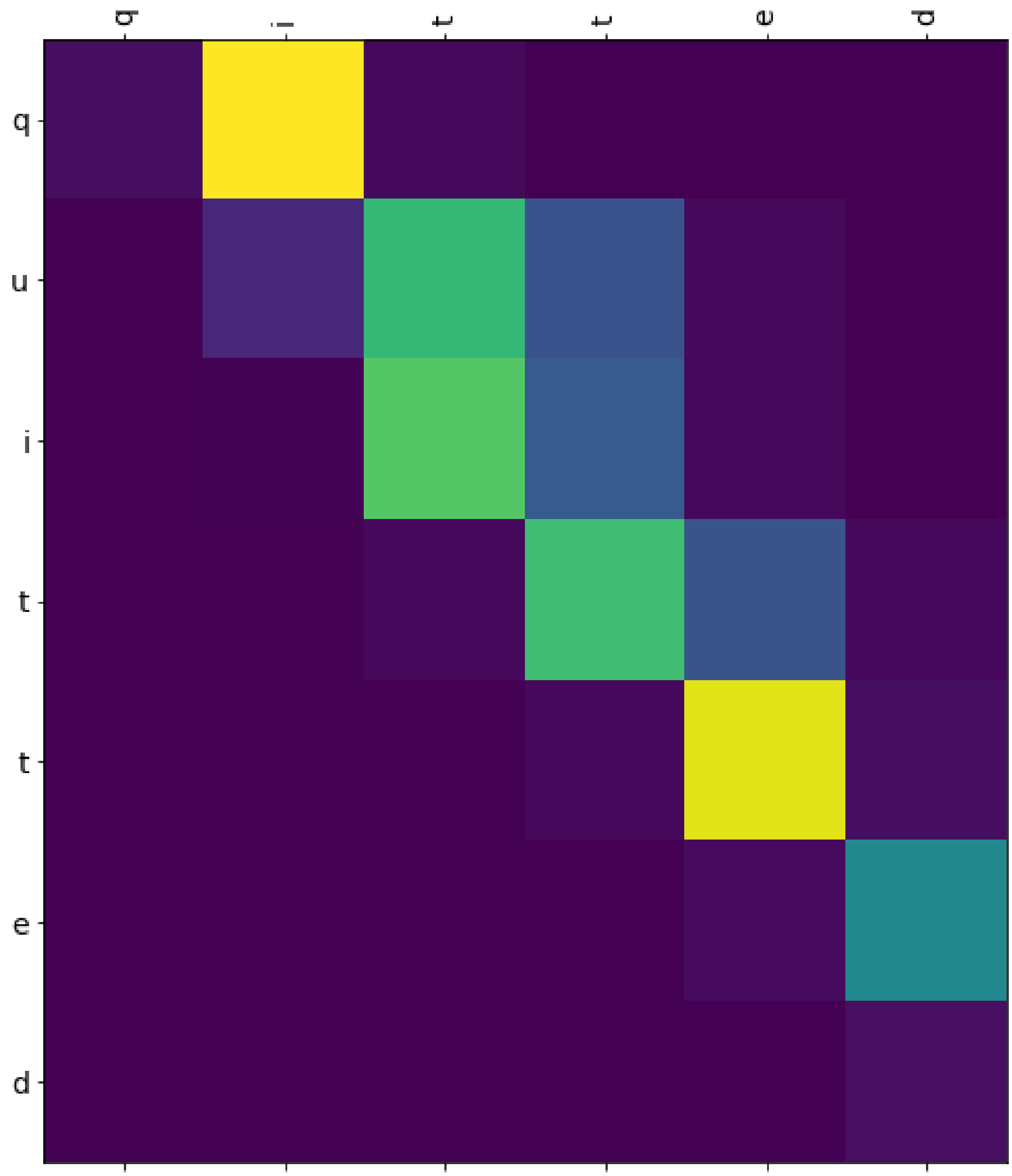
[54]:
```
sentence = unigram_train['input'].values[5]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
 ↪gram = 'uni')

print('input : ', sentence)
```

```
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[5])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input: qitted
predicted output: quitted
actual output: quitted
```

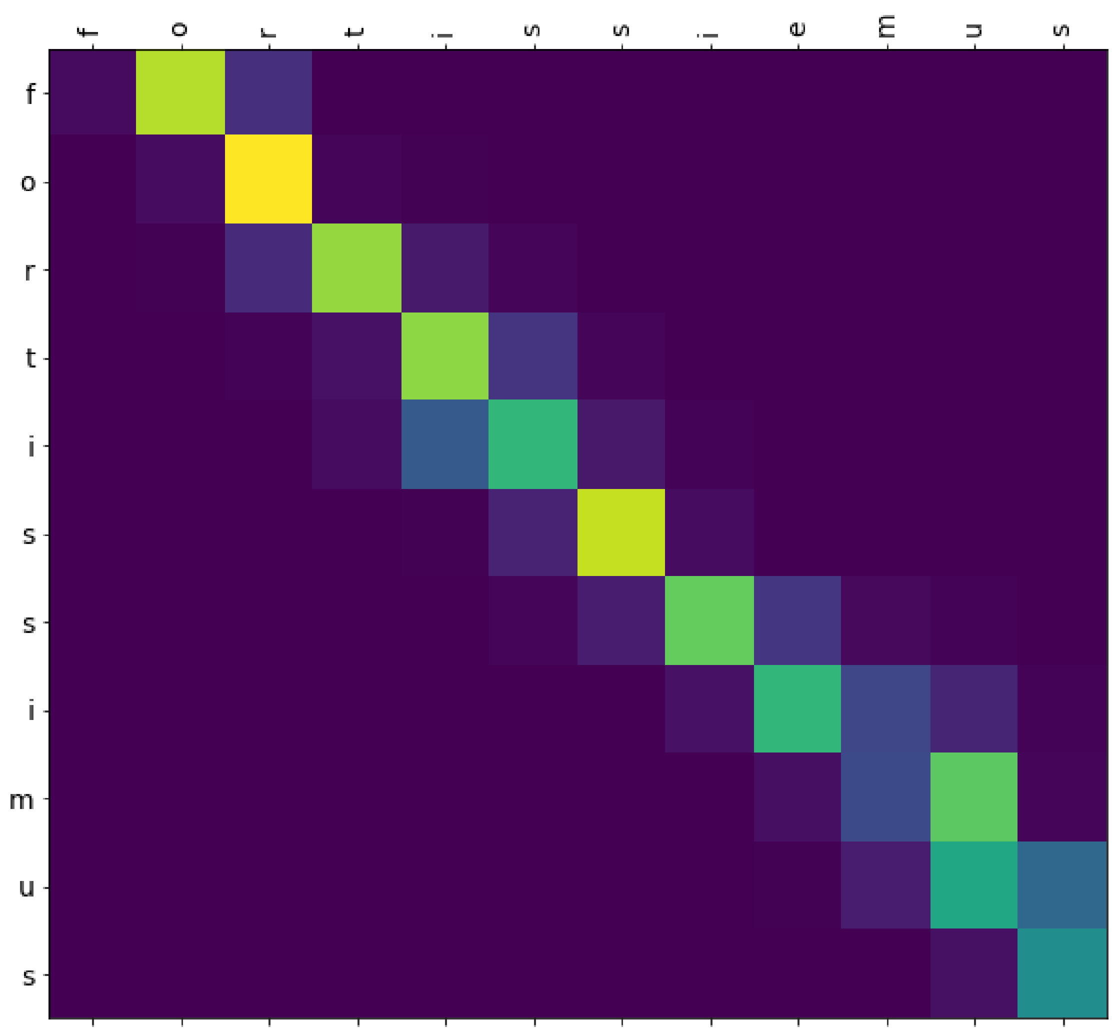


[61]:
```
sentence = unigram_train['input'].values[11]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
↪gram = 'uni')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[11])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input: fortissiemus
predicted output: fortissimus
actual output: fortissimus
```

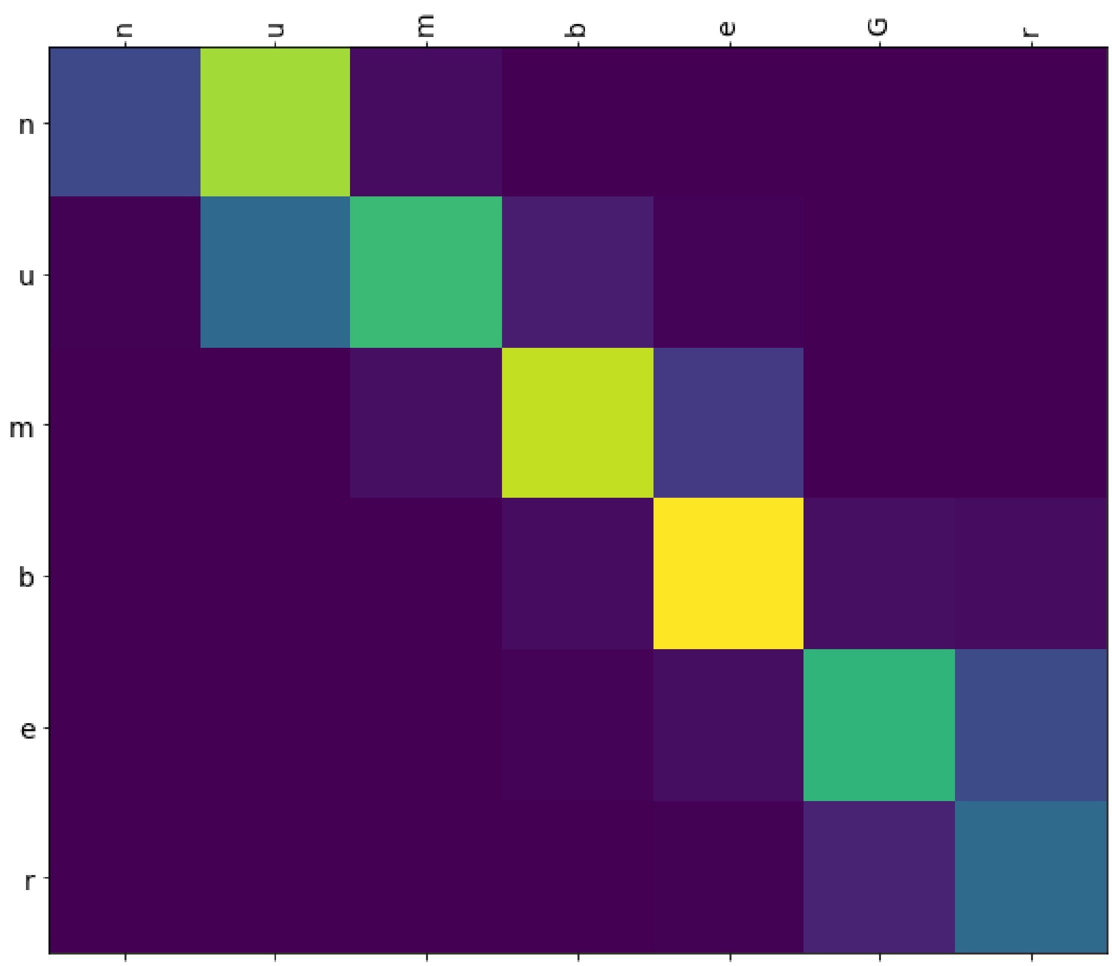


[65]:
```
sentence = unigram_train['input'].values[14]
result, attention_plot = predict(sentence, unigram_vec, unigram_index_to_word,␣
 ↪gram = 'uni')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', unigram_train['output'].values[14])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input: numbeGr
predicted output: number
actual output: number
```



[47]:
```
val_bleu = 0
for i in tqdm(range(unigram_val.shape[0])):
```

```
    inp = unigram_val['input'].values[i]
    out = unigram_val['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in tqdm(range(unigram_train.shape[0])):
    inp = unigram_train['input'].values[i];
    out = unigram_train['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in tqdm(range(unigram_test.shape[0])):
    inp = unigram_test['input'].values[i]
    out = unigram_test['output'].values[i]
    pred, _ = predict(inp, unigram_vec, unigram_index_to_word, gram = 'uni')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/unigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/unigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/unigram_test.shape[0])
```

```
100%|
  | 3678/3678 [04:14<00:00, 14.46it/s]
100%|
 | 33101/33101 [38:27<00:00, 14.35it/s]
100%|
  | 3150/3150 [03:38<00:00, 14.41it/s]

BLEU Score on train:  0.8553528187207379
BLEU Score on val:  0.8027546198959468
BLEU Score on test:  0.712202060966304
```

3.2 BiGram

[107]:
```
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 26
```

[104]:
```
model = encoder_decoder(vocab_size, embedding_dim, lstm_size, lstm_size,␣
 ↪maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)

callbacks = [ModelCheckpoint('concat_best_bigram.h5', save_best_only= True,␣
 ↪verbose = 1),
```

```
            EarlyStopping(patience = 5, verbose = 1),
            ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = bigram_train_dataset,
          steps_per_epoch = bigram_train.shape[0]//batch_size,
          validation_data = bigram_val_dataset,
          validation_steps = bigram_val.shape[0]//batch_size,
          epochs = 100,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/100
6910/6910 [==============================] - ETA: 0s - loss: 0.1299
Epoch 00001: val_loss improved from inf to 0.04310, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 847s 123ms/step - loss: 0.1299 -
val_loss: 0.0431
Epoch 2/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0464
Epoch 00002: val_loss improved from 0.04310 to 0.03256, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 869s 126ms/step - loss: 0.0464 -
val_loss: 0.0326
Epoch 3/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0377
Epoch 00003: val_loss improved from 0.03256 to 0.02836, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 878s 127ms/step - loss: 0.0377 -
val_loss: 0.0284
Epoch 4/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0334
Epoch 00004: val_loss improved from 0.02836 to 0.02540, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 890s 129ms/step - loss: 0.0334 -
val_loss: 0.0254
Epoch 5/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0305
Epoch 00005: val_loss improved from 0.02540 to 0.02419, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 877s 127ms/step - loss: 0.0305 -
val_loss: 0.0242
Epoch 6/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0285
Epoch 00006: val_loss improved from 0.02419 to 0.02344, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 864s 125ms/step - loss: 0.0285 -
val_loss: 0.0234
Epoch 7/100
```

```
6910/6910 [==============================] - ETA: 0s - loss: 0.0270
Epoch 00007: val_loss improved from 0.02344 to 0.02228, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 904s 131ms/step - loss: 0.0270 -
val_loss: 0.0223
Epoch 8/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0260
Epoch 00008: val_loss improved from 0.02228 to 0.02181, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 934s 135ms/step - loss: 0.0260 -
val_loss: 0.0218
Epoch 9/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0250
Epoch 00009: val_loss improved from 0.02181 to 0.02144, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 922s 133ms/step - loss: 0.0250 -
val_loss: 0.0214
Epoch 10/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0243
Epoch 00010: val_loss improved from 0.02144 to 0.02106, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 884s 128ms/step - loss: 0.0243 -
val_loss: 0.0211
Epoch 11/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0235
Epoch 00011: val_loss improved from 0.02106 to 0.02065, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 871s 126ms/step - loss: 0.0235 -
val_loss: 0.0207
Epoch 12/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0229
Epoch 00012: val_loss did not improve from 0.02065
6910/6910 [==============================] - 874s 126ms/step - loss: 0.0229 -
val_loss: 0.0207
Epoch 13/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0223
Epoch 00013: val_loss improved from 0.02065 to 0.02025, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 978s 141ms/step - loss: 0.0223 -
val_loss: 0.0203
Epoch 14/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0218
Epoch 00014: val_loss improved from 0.02025 to 0.02014, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 836s 121ms/step - loss: 0.0218 -
val_loss: 0.0201
Epoch 15/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0215
```

```
Epoch 00015: val_loss did not improve from 0.02014
6910/6910 [==============================] - 853s 123ms/step - loss: 0.0215 -
val_loss: 0.0202
Epoch 16/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0210
Epoch 00016: val_loss improved from 0.02014 to 0.01981, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 881s 128ms/step - loss: 0.0210 -
val_loss: 0.0198
Epoch 17/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0206
Epoch 00017: val_loss improved from 0.01981 to 0.01966, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 849s 123ms/step - loss: 0.0206 -
val_loss: 0.0197
Epoch 18/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0203
Epoch 00018: val_loss improved from 0.01966 to 0.01951, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 830s 120ms/step - loss: 0.0203 -
val_loss: 0.0195
Epoch 19/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0201
Epoch 00019: val_loss did not improve from 0.01951
6910/6910 [==============================] - 832s 120ms/step - loss: 0.0201 -
val_loss: 0.0195
Epoch 20/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0197
Epoch 00020: val_loss improved from 0.01951 to 0.01923, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 816s 118ms/step - loss: 0.0197 -
val_loss: 0.0192
Epoch 21/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0195
Epoch 00021: val_loss did not improve from 0.01923
6910/6910 [==============================] - 815s 118ms/step - loss: 0.0195 -
val_loss: 0.0194
Epoch 22/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0191
Epoch 00022: val_loss did not improve from 0.01923
6910/6910 [==============================] - 846s 122ms/step - loss: 0.0191 -
val_loss: 0.0193
Epoch 23/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0190
Epoch 00023: val_loss improved from 0.01923 to 0.01905, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 844s 122ms/step - loss: 0.0190 -
val_loss: 0.0191
```

```
Epoch 24/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0187
Epoch 00024: val_loss did not improve from 0.01905
6910/6910 [==============================] - 843s 122ms/step - loss: 0.0187 -
val_loss: 0.0196
Epoch 25/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0184
Epoch 00025: val_loss improved from 0.01905 to 0.01900, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 866s 125ms/step - loss: 0.0184 -
val_loss: 0.0190
Epoch 26/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0182
Epoch 00026: val_loss did not improve from 0.01900

Epoch 00026: ReduceLROnPlateau reducing learning rate to 0.0001000000474974513.
6910/6910 [==============================] - 949s 137ms/step - loss: 0.0182 -
val_loss: 0.0191
Epoch 27/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0159
Epoch 00027: val_loss improved from 0.01900 to 0.01819, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 881s 128ms/step - loss: 0.0159 -
val_loss: 0.0182
Epoch 28/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0148
Epoch 00028: val_loss improved from 0.01819 to 0.01799, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 836s 121ms/step - loss: 0.0148 -
val_loss: 0.0180
Epoch 29/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0143
Epoch 00029: val_loss improved from 0.01799 to 0.01787, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 834s 121ms/step - loss: 0.0143 -
val_loss: 0.0179
Epoch 30/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0139
Epoch 00030: val_loss improved from 0.01787 to 0.01779, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 832s 120ms/step - loss: 0.0139 -
val_loss: 0.0178
Epoch 31/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0136
Epoch 00031: val_loss improved from 0.01779 to 0.01773, saving model to
concat_best_bigram.h5
6910/6910 [==============================] - 828s 120ms/step - loss: 0.0136 -
val_loss: 0.0177
```

```
Epoch 32/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0134
Epoch 00032: val_loss did not improve from 0.01773
6910/6910 [==============================] - 829s 120ms/step - loss: 0.0134 -
val_loss: 0.0178
Epoch 33/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0132
Epoch 00033: val_loss did not improve from 0.01773
6910/6910 [==============================] - 820s 119ms/step - loss: 0.0132 -
val_loss: 0.0178
Epoch 34/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0130
Epoch 00034: val_loss did not improve from 0.01773

Epoch 00034: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 820s 119ms/step - loss: 0.0130 -
val_loss: 0.0178
Epoch 35/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0127
Epoch 00035: val_loss did not improve from 0.01773
6910/6910 [==============================] - 819s 119ms/step - loss: 0.0127 -
val_loss: 0.0178
Epoch 36/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0126
Epoch 00036: val_loss did not improve from 0.01773
6910/6910 [==============================] - 820s 119ms/step - loss: 0.0126 -
val_loss: 0.0177
Epoch 00036: early stopping
```

[104]:
```
<tensorflow.python.keras.callbacks.History at 0x22706ba1ac8>
```

[108]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, lstm_size, maxlen, maxlen, 'concat', att_units,␣
 ↪bigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('concat_best_bigram.h5')
```

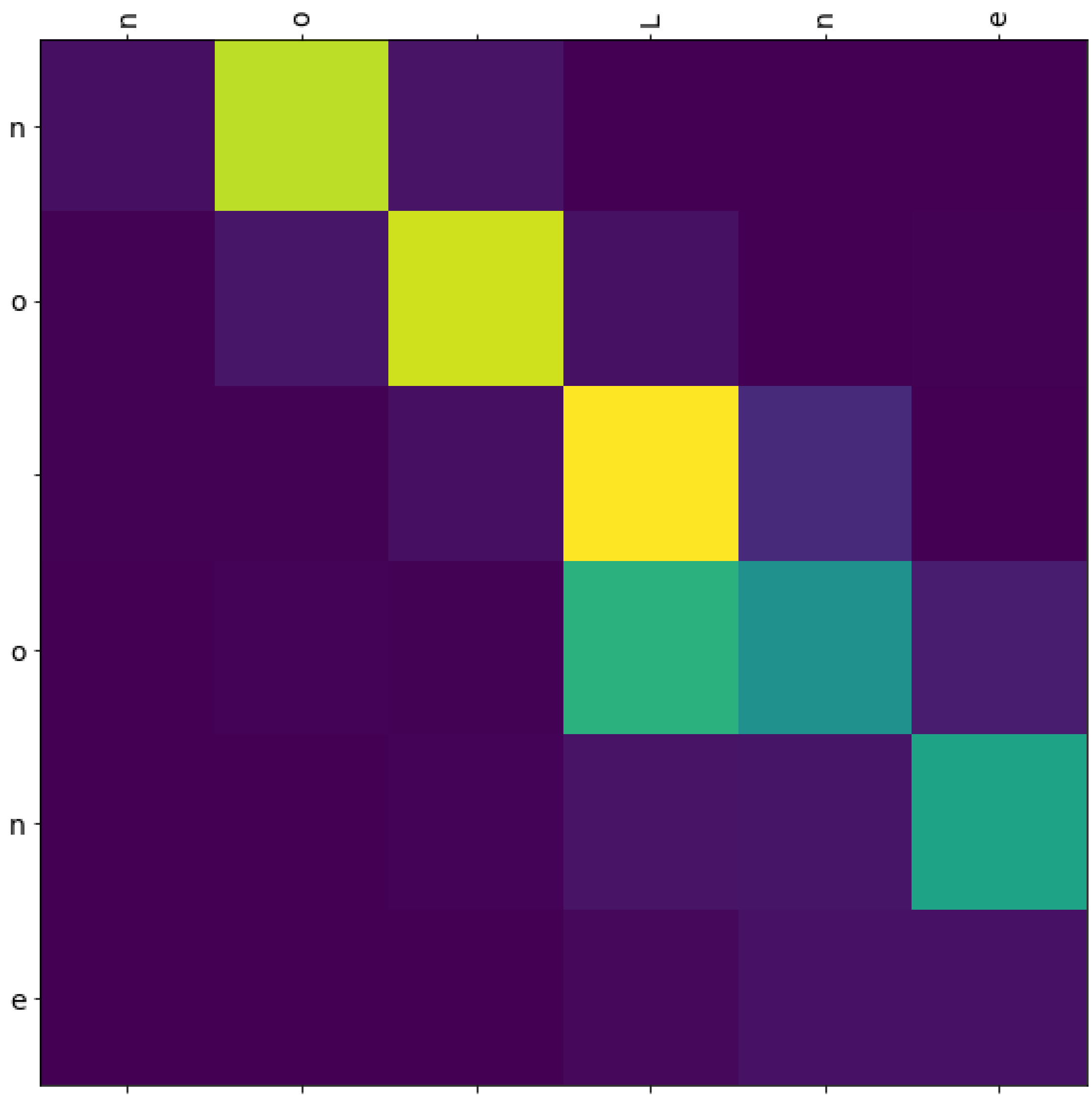
[109]:
```
sentence = bigram_train['input'].values[4]
result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
 ↪gram = 'bi')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[4])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
```

```
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  no Lne
predicted output :  no one
actual output : no one
```

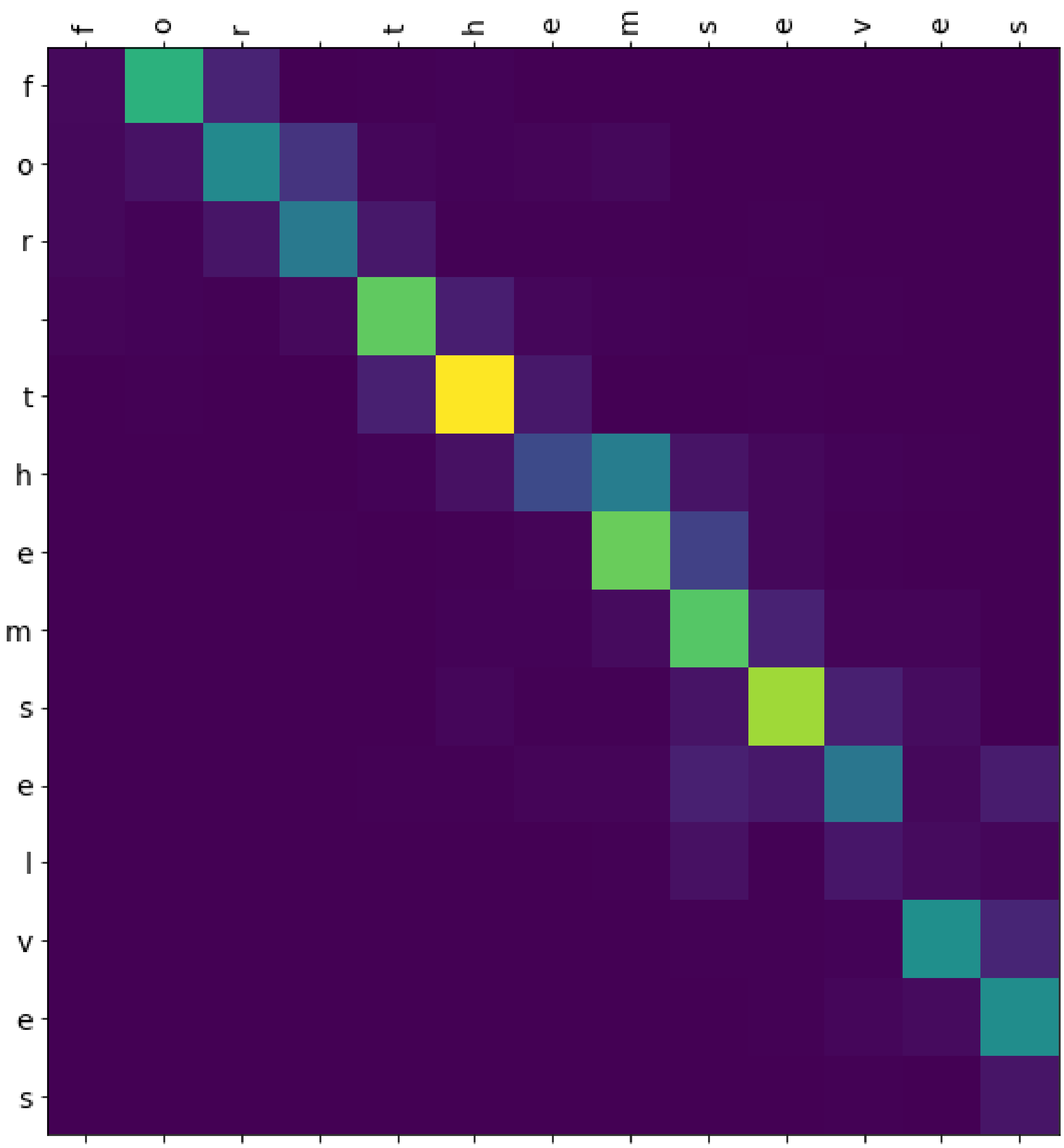



```
[110]: sentence = bigram_train['input'].values[6]
result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
 ↪gram = 'bi')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[6])
```

```
attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  for themseves
predicted output :  for themselves
actual output : for themselves
```

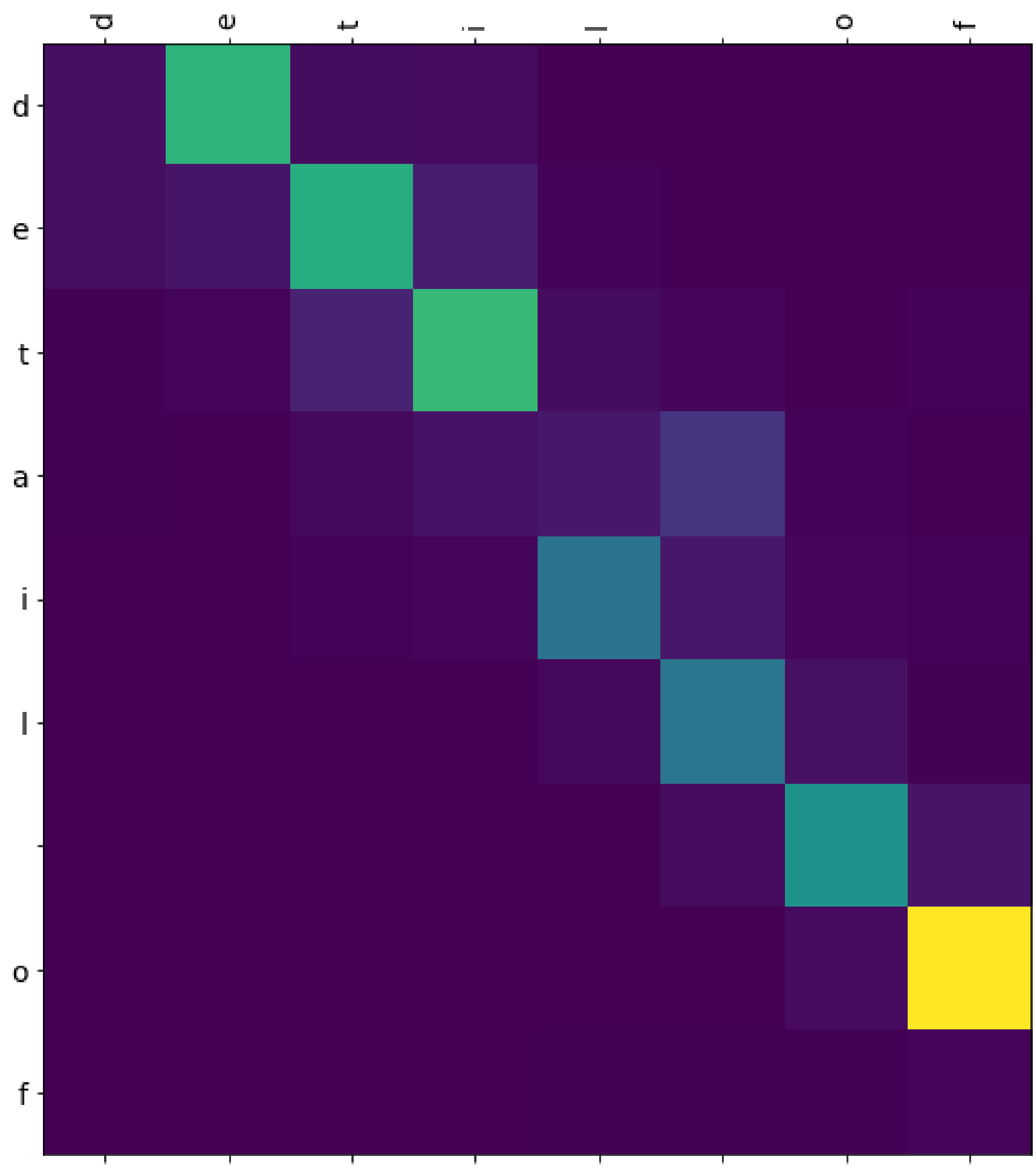


[111]:
```
sentence = bigram_train['input'].values[7]
result, attention_plot = predict(sentence, bigram_vec, bigram_index_to_word,␣
 ↪gram = 'bi')
```

```
print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', bigram_train['output'].values[7])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  detil of
predicted output :  detail of
actual output : detail of
```



[7]:
```
val_bleu = 0
for i in range(bigram_val.shape[0]):
    inp = bigram_val['input'].values[i]
    out = bigram_val['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in range(bigram_train.shape[0]):
    inp = bigram_train['input'].values[i];
    out = bigram_train['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in range(bigram_test.shape[0]):
    inp = bigram_test['input'].values[i]
    out = bigram_test['output'].values[i]
    pred, _ = predict(inp, bigram_vec, bigram_index_to_word, gram = 'bi')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/bigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/bigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/bigram_test.shape[0])
```

```
BLEU Score on train:  0.9808526374708109
BLEU Score on val:  0.9669617055111845
BLEU Score on test:  0.9539630640021209
```

3.3 TriGram

[112]:
```
lstm_size = 256
embedding_dim = 100
att_units = 256
maxlen = 34
```

[114]:
```
model = encoder_decoder(vocab_size, embedding_dim, lstm_size, lstm_size,␣
 ↪maxlen, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)


callbacks = [ModelCheckpoint('concat_best_trigram.h5', save_best_only= True,␣
 ↪verbose = 1),
             EarlyStopping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]
```

```
model.fit(x = trigram_train_dataset,
          steps_per_epoch = trigram_train.shape[0]//batch_size,
          validation_data = trigram_val_dataset,
          validation_steps = trigram_val.shape[0]//batch_size,
          epochs = 100,
          verbose = 1,
          callbacks = callbacks)
```

```
Epoch 1/100
6910/6910 [==============================] - ETA: 0s - loss: 0.1372
Epoch 00001: val_loss improved from inf to 0.04223, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1156s 167ms/step - loss: 0.1372 -
val_loss: 0.0422
Epoch 2/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0439
Epoch 00002: val_loss improved from 0.04223 to 0.03123, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1154s 167ms/step - loss: 0.0439 -
val_loss: 0.0312
Epoch 3/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0349
Epoch 00003: val_loss improved from 0.03123 to 0.02453, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1158s 168ms/step - loss: 0.0349 -
val_loss: 0.0245
Epoch 4/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0303
Epoch 00004: val_loss improved from 0.02453 to 0.02350, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1159s 168ms/step - loss: 0.0303 -
val_loss: 0.0235
Epoch 5/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0277
Epoch 00005: val_loss improved from 0.02350 to 0.02035, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1163s 168ms/step - loss: 0.0277 -
val_loss: 0.0204
Epoch 6/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0256
Epoch 00006: val_loss improved from 0.02035 to 0.01934, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1158s 168ms/step - loss: 0.0256 -
val_loss: 0.0193
Epoch 7/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0241
Epoch 00007: val_loss improved from 0.01934 to 0.01881, saving model to
```

```
concat_best_trigram.h5
6910/6910 [==============================] - 1157s 167ms/step - loss: 0.0241 -
val_loss: 0.0188
Epoch 8/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0229
Epoch 00008: val_loss improved from 0.01881 to 0.01879, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1162s 168ms/step - loss: 0.0229 -
val_loss: 0.0188
Epoch 9/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0219
Epoch 00009: val_loss improved from 0.01879 to 0.01782, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1174s 170ms/step - loss: 0.0219 -
val_loss: 0.0178
Epoch 10/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0215
Epoch 00010: val_loss improved from 0.01782 to 0.01715, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1185s 171ms/step - loss: 0.0215 -
val_loss: 0.0171
Epoch 11/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0206
Epoch 00011: val_loss improved from 0.01715 to 0.01670, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1195s 173ms/step - loss: 0.0206 -
val_loss: 0.0167
Epoch 12/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0200
Epoch 00012: val_loss improved from 0.01670 to 0.01650, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1195s 173ms/step - loss: 0.0200 -
val_loss: 0.0165
Epoch 13/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0194
Epoch 00013: val_loss improved from 0.01650 to 0.01627, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1182s 171ms/step - loss: 0.0194 -
val_loss: 0.0163
Epoch 14/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0189
Epoch 00014: val_loss improved from 0.01627 to 0.01613, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1188s 172ms/step - loss: 0.0189 -
val_loss: 0.0161
Epoch 15/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0186
Epoch 00015: val_loss improved from 0.01613 to 0.01572, saving model to
```

```
concat_best_trigram.h5
6910/6910 [==============================] - 1218s 176ms/step - loss: 0.0186 -
val_loss: 0.0157
Epoch 16/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0182
Epoch 00016: val_loss improved from 0.01572 to 0.01561, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1225s 177ms/step - loss: 0.0182 -
val_loss: 0.0156
Epoch 17/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0179
Epoch 00017: val_loss did not improve from 0.01561
6910/6910 [==============================] - 1229s 178ms/step - loss: 0.0179 -
val_loss: 0.0164
Epoch 18/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0175
Epoch 00018: val_loss improved from 0.01561 to 0.01544, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1236s 179ms/step - loss: 0.0175 -
val_loss: 0.0154
Epoch 19/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0172
Epoch 00019: val_loss improved from 0.01544 to 0.01532, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1295s 187ms/step - loss: 0.0172 -
val_loss: 0.0153
Epoch 20/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0168
Epoch 00020: val_loss improved from 0.01532 to 0.01530, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1249s 181ms/step - loss: 0.0168 -
val_loss: 0.0153
Epoch 21/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0168
Epoch 00021: val_loss improved from 0.01530 to 0.01508, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1230s 178ms/step - loss: 0.0168 -
val_loss: 0.0151
Epoch 22/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0164
Epoch 00022: val_loss improved from 0.01508 to 0.01506, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1227s 178ms/step - loss: 0.0164 -
val_loss: 0.0151
Epoch 23/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0163
Epoch 00023: val_loss did not improve from 0.01506
6910/6910 [==============================] - 1206s 175ms/step - loss: 0.0163 -
```

```
val_loss: 0.0151
Epoch 24/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0160
Epoch 00024: val_loss did not improve from 0.01506

Epoch 00024: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
6910/6910 [==============================] - 1206s 175ms/step - loss: 0.0160 -
val_loss: 0.0158
Epoch 25/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0146
Epoch 00025: val_loss improved from 0.01506 to 0.01435, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1221s 177ms/step - loss: 0.0146 -
val_loss: 0.0144
Epoch 26/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0133
Epoch 00026: val_loss improved from 0.01435 to 0.01406, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1249s 181ms/step - loss: 0.0133 -
val_loss: 0.0141
Epoch 27/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0128
Epoch 00027: val_loss improved from 0.01406 to 0.01390, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1269s 184ms/step - loss: 0.0128 -
val_loss: 0.0139
Epoch 28/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0124
Epoch 00028: val_loss improved from 0.01390 to 0.01382, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1301s 188ms/step - loss: 0.0124 -
val_loss: 0.0138
Epoch 29/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0122
Epoch 00029: val_loss improved from 0.01382 to 0.01374, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1398s 202ms/step - loss: 0.0122 -
val_loss: 0.0137
Epoch 30/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0119
Epoch 00030: val_loss improved from 0.01374 to 0.01366, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1362s 197ms/step - loss: 0.0119 -
val_loss: 0.0137
Epoch 31/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0117
Epoch 00031: val_loss improved from 0.01366 to 0.01364, saving model to
concat_best_trigram.h5
```

```
6910/6910 [==============================] - 1326s 192ms/step - loss: 0.0117 -
val_loss: 0.0136
Epoch 32/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0116
Epoch 00032: val_loss did not improve from 0.01364
6910/6910 [==============================] - 1265s 183ms/step - loss: 0.0116 -
val_loss: 0.0136
Epoch 33/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0114
Epoch 00033: val_loss improved from 0.01364 to 0.01358, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1266s 183ms/step - loss: 0.0114 -
val_loss: 0.0136
Epoch 34/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0113
Epoch 00034: val_loss improved from 0.01358 to 0.01357, saving model to
concat_best_trigram.h5

Epoch 00034: ReduceLROnPlateau reducing learning rate to 1.0000000474974514e-05.
6910/6910 [==============================] - 1287s 186ms/step - loss: 0.0113 -
val_loss: 0.0136
Epoch 35/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0111
Epoch 00035: val_loss improved from 0.01357 to 0.01353, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1298s 188ms/step - loss: 0.0111 -
val_loss: 0.0135
Epoch 36/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0110
Epoch 00036: val_loss improved from 0.01353 to 0.01350, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1385s 200ms/step - loss: 0.0110 -
val_loss: 0.0135
Epoch 37/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0110
Epoch 00037: val_loss improved from 0.01350 to 0.01350, saving model to
concat_best_trigram.h5
6910/6910 [==============================] - 1369s 198ms/step - loss: 0.0110 -
val_loss: 0.0135
Epoch 38/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00038: val_loss improved from 0.01350 to 0.01349, saving model to
concat_best_trigram.h5

Epoch 00038: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
6910/6910 [==============================] - 1333s 193ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 39/100
```

```
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00039: val_loss did not improve from 0.01349
6910/6910 [==============================] - 1392s 201ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 40/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00040: val_loss did not improve from 0.01349
6910/6910 [==============================] - 1350s 195ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 41/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00041: val_loss did not improve from 0.01349

Epoch 00041: ReduceLROnPlateau reducing learning rate to 1.0000001111620805e-07.
6910/6910 [==============================] - 1396s 202ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 42/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00042: val_loss did not improve from 0.01349
6910/6910 [==============================] - 1398s 202ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 43/100
6910/6910 [==============================] - ETA: 0s - loss: 0.0109
Epoch 00043: val_loss did not improve from 0.01349
6910/6910 [==============================] - 1371s 198ms/step - loss: 0.0109 -
val_loss: 0.0135
Epoch 00043: early stopping
```

[114]:
```
<tensorflow.python.keras.callbacks.History at 0x226efe79bc8>
```

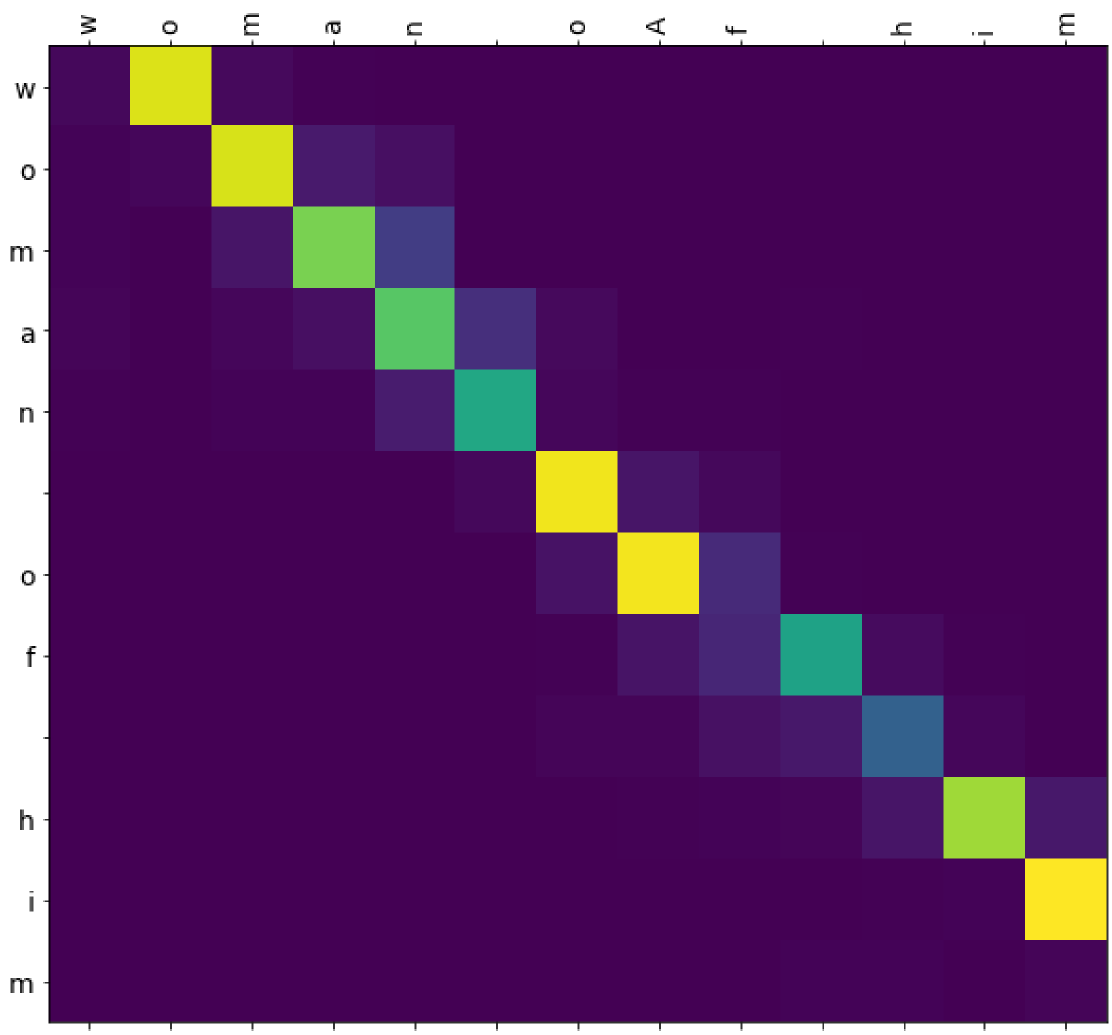
[115]:
```
pred_model = pred_Encoder_decoder(vocab_size, vocab_size, embedding_dim,␣
 ↪lstm_size, lstm_size, maxlen, maxlen, 'concat', att_units,␣
 ↪trigram_word_to_index)
pred_model.compile(optimizer = 'Adam', loss = loss_function)
pred_model.build(input_shape= (None, 1, maxlen))
pred_model.load_weights('concat_best_trigram.h5')
```

[116]:
```
sentence = trigram_train['input'].values[4]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[4])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  woman oAf him
predicted output :  woman of him
actual output : woman of him
```



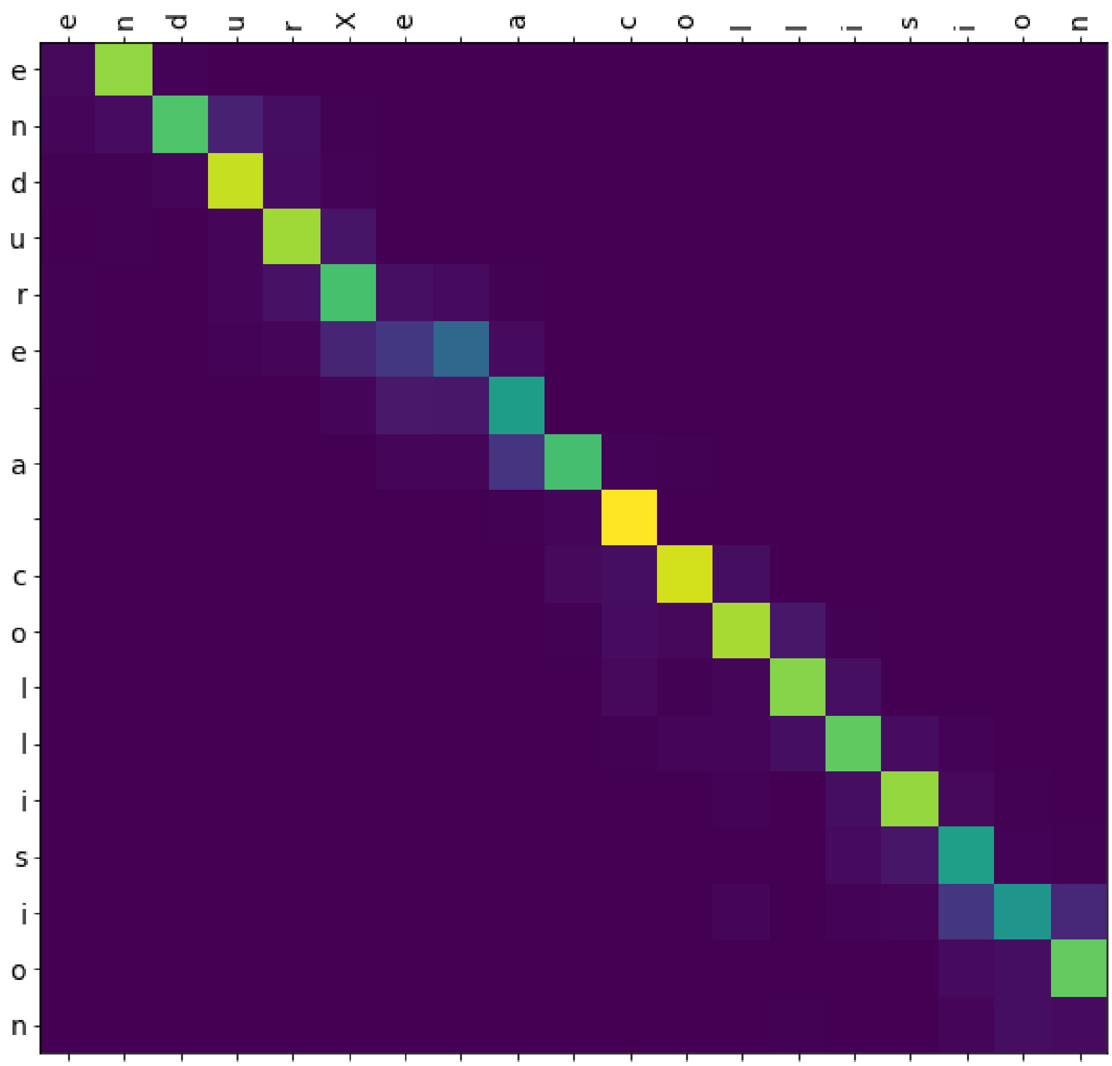
[117]:
```python
sentence = trigram_train['input'].values[7]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[7])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  endurXe a collision
```

```
predicted output :  endure a collision
actual output : endure a collision
```



[118]:

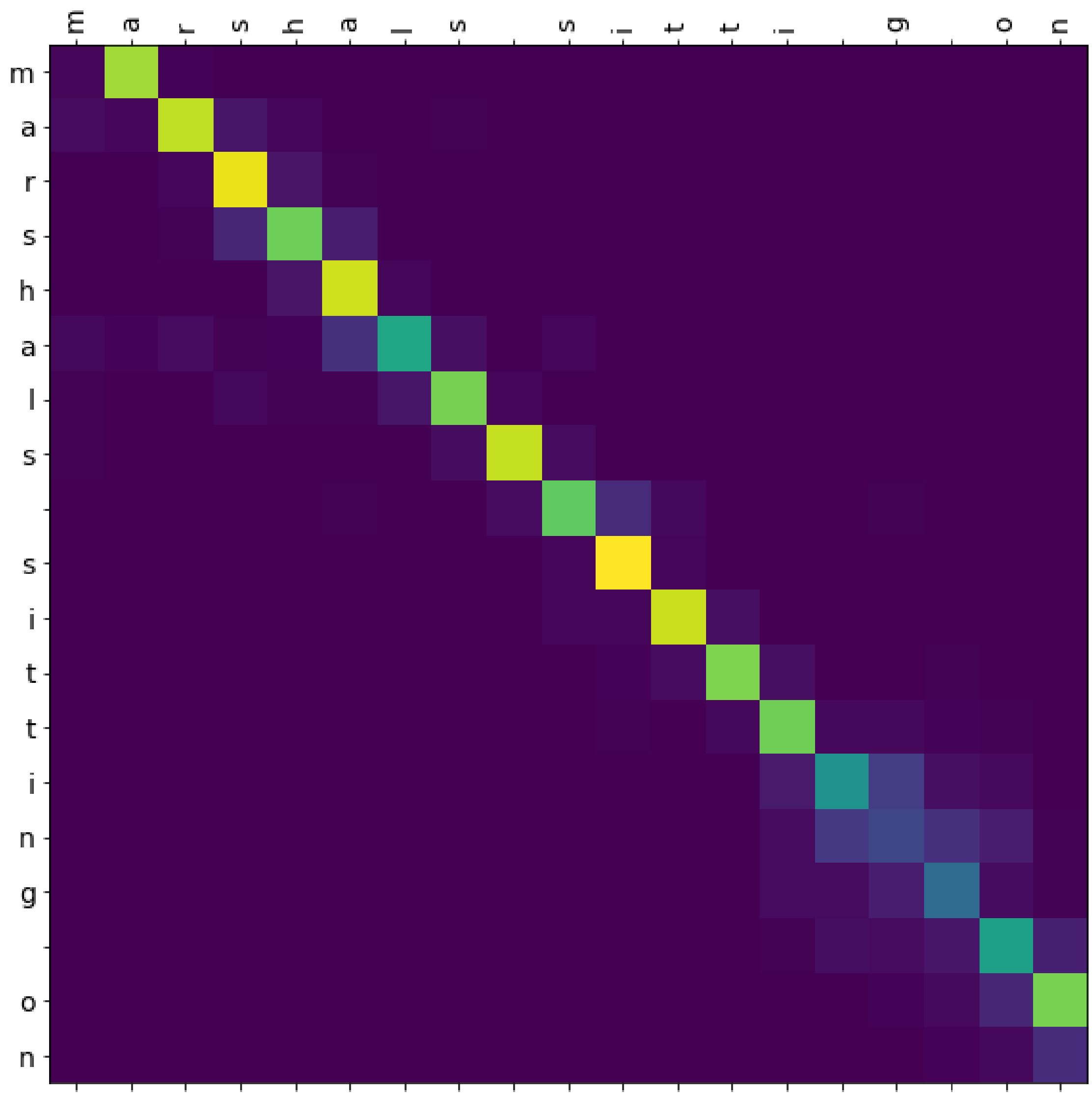
```
sentence = trigram_train['input'].values[10]
result, attention_plot = predict(sentence, trigram_vec, trigram_index_to_word,␣
 ↪gram = 'tri')

print('input : ', sentence)
print('predicted output : ',result)
print('actual output :', trigram_train['output'].values[10])

attention_plot = attention_plot[:len(list(result)), :len(list(sentence))]
plot_attention(attention_plot, list(sentence), list(result))
```

```
input :  marshals sitti g on
```

```
predicted output :  marshals sitting on
actual output : marshals sitting on
```




```
[8]: val_bleu = 0
for i in range(trigram_val.shape[0]):
    inp = trigram_val['input'].values[i]
    out = trigram_val['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    val_bleu += sentence_bleu([out], pred)

train_bleu = 0
for i in range(trigram_train.shape[0]):
    inp = trigram_train['input'].values[i];
```

```
    out = trigram_train['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    train_bleu += sentence_bleu([out], pred)

test_bleu = 0
for i in range(trigram_test.shape[0]):
    inp = trigram_test['input'].values[i]
    out = trigram_test['output'].values[i]
    pred, _ = predict(inp, trigram_vec, trigram_index_to_word, gram = 'tri')
    test_bleu += sentence_bleu([out], pred)

print('BLEU Score on train: ',train_bleu/trigram_train.shape[0])
print('BLEU Score on val: ',val_bleu/trigram_val.shape[0])
print('BLEU Score on test: ',test_bleu/trigram_test.shape[0])
```

```
BLEU Score on train:  0.9889611211686458
BLEU Score on val:  0.9813112757412255
BLEU Score on test:  0.9693013446155896
```

3 Conclusion

[9]:
```
from prettytable import PrettyTable

myTable = PrettyTable(["n-gram", "Model Name", "Train BLEU Score", "Val BLEU␣
↪Score", "Test BLEU Score"])
myTable.add_row(["1-gram", "Seq2Seq", "0.834", "0.746", "0.684"])
myTable.add_row(["2-gram", "Seq2Seq", "0.965", "0.947", "0.932"])
myTable.add_row(["3-gram", "Seq2Seq", "0.968", "0.958", "0.949"])

myTable.add_row(["1-gram", "Seq2Seq with Attention Mechanism", "0.869", "0.
↪792", "0.707"])
myTable.add_row(["2-gram", "Seq2Seq with Attention Mechanism", "0.972", "0.
↪959", "0.946"])
myTable.add_row(["3-gram", "Seq2Seq with Attention Mechanism", "0.981", "0.
↪974", "0.962"])

myTable.add_row(["1-gram", "Bi-directional Seq2Seq with Attention Mechanism",␣
↪"0.855", "0.802", "0.712"])
myTable.add_row(["2-gram", "Bi-directional Seq2Seq with Attention Mechanism",␣
↪"0.980", "0.967", "0.954"])
myTable.add_row(["3-gram", "Bi-directional Seq2Seq with Attention Mechanism",␣
↪"0.989", "0.981", "0.969"])

print(myTable)
```

```
+--------+-------------------------------------------------+------------------+-
-----------------+-----------------+
```

n-gram	Model Name	Train BLEU Score	Val BLEU Score	Test BLEU Score
1-gram	Seq2Seq	0.834	0.746	0.684
2-gram	Seq2Seq	0.965	0.947	0.932
3-gram	Seq2Seq	0.968	0.958	0.949
1-gram	Seq2Seq with Attention Mechanism	0.869	0.792	0.707
2-gram	Seq2Seq with Attention Mechanism	0.972	0.959	0.946
3-gram	Seq2Seq with Attention Mechanism	0.981	0.974	0.962
1-gram	Bi-directional Seq2Seq with Attention Mechanism	0.855	0.802	0.712
2-gram	Bi-directional Seq2Seq with Attention Mechanism	0.980	0.967	0.954
3-gram	Bi-directional Seq2Seq with Attention Mechanism	0.989	0.981	0.969