Machine Translation using sequence 2 sequence model

by LinguaLinkers

P23DS015-Neha Patel
P23DS028-Adwitiya Gaurav

Content Table

Machine Translation

Machine translation is the process of automatically translating text or speech from one natural language to another using computational algorithms and models. The goal of machine translation is to enable communication and understanding between people who speak different languages without the need for human translators.

1. **Data Collection**: Machine translation systems require large amounts of parallel text data, known as parallel corpora, consisting of sentences or documents in multiple languages and their translations.
2. **Preprocessing**: The collected data is preprocessed to clean and tokenize the text, handle punctuation, remove special characters, and normalize the text to ensure consistency.
3. **Model Selection**: Machine translation models can be categorized into different types based on their underlying algorithms such as rule based, statistical, neural machine translation.
4. **Inference**: Once trained, the machine translation model can be used to translate new input text from the source language to the target language. During inference, the model processes the input text using its learned parameters and generates the corresponding translation.
5. **Evaluation**: The quality of machine translation systems is evaluated using metrics such as BLEU (Bilingual Evaluation Understudy), METEOR (Metric for Evaluation of Translation with Explicit Ordering), TER (Translation Edit Rate), and human evaluations. These metrics measure the similarity between the machine-generated translations and human reference translations.

Sequence 2 sequence learning

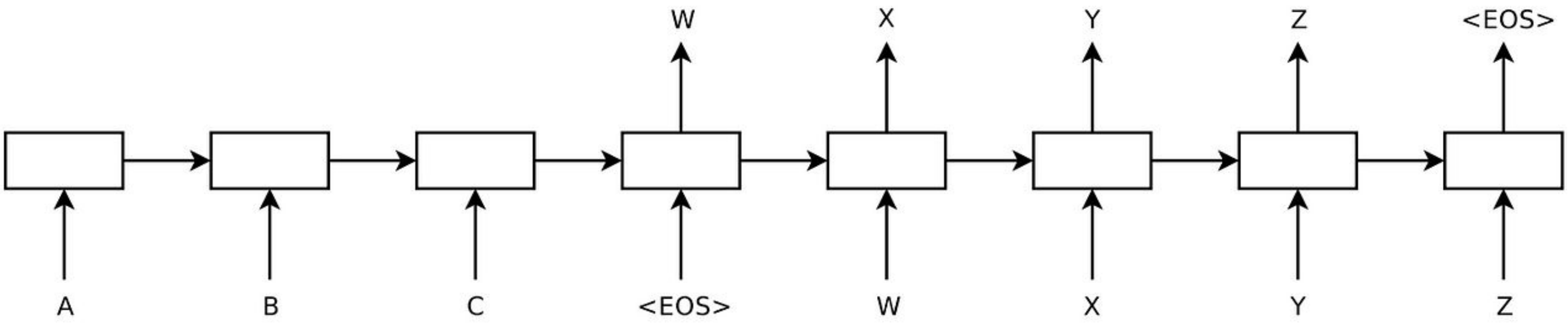
Sequence-to-sequence learning (Seq2Seq) is about training models to convert sequences from one domain (e.g. sentences in English) to sequences in another domain (e.g. the same sentences marathi).

This can be used for machine translation or for free-from question answering (generating a natural language answer given a natural language question) -- in general, it is applicable any time you need to generate text.

There are multiple ways to handle this task, either using RNNs or using 1D convnets. Here we will focus on RNNs.

In the general case, input sequences and output sequences have different lengths (e.g. machine translation) and the entire input sequence is required in order to start predicting the target. This requires a more advanced setup, which is what people commonly refer to when mentioning "sequence to sequence models" with no further context. Here's how it works:

- A RNN layer (or stack thereof) acts as "encoder": it processes the input sequence and returns its own internal state. Note that we discard the outputs of the encoder RNN, only recovering the state. This state will serve as the "context", or "conditioning", of the decoder in the next step.
- Another RNN layer (or stack thereof) acts as "decoder": it is trained to predict the next characters of the target sequence, given previous characters of the target sequence. Specifically, it is trained to turn the target sequences into the same sequences but offset by one timestep in the future, a training process called "teacher forcing" in this context. Importantly, the encoder uses as initial state the state vectors from the encoder, which is how the decoder obtains information about what it is supposed to generate. Effectively, the decoder learns to generate targets[t+1...] given targets[...t], *conditioned on the input sequence*.


W
X
Y
Z
<EOS>
A
B
C
<EOS>
W
X
Y
Z

In inference mode, i.e. when we want to decode unknown input sequences, we go through a slightly different process:

- 1) Encode the input sequence into state vectors.
- 2) Start with a target sequence of size 1 (just the start-of-sequence character).
- 3) Feed the state vectors and 1-char target sequence to the decoder to produce predictions for the next character.
- 4) Sample the next character using these predictions (we simply use argmax).
- 5) Append the sampled character to the target sequence
- 6) Repeat until we generate the end-of-sequence character or we hit the character limit.

Implementing a basic seq2seq model for english to marathi translation using keras

1. **Data Preparation**:
    - Gather a parallel corpus containing English sentences and their corresponding translations in Marathi.
    - Preprocess the data by tokenizing the sentences, converting them into sequences of integers, and padding them to ensure they all have the same length.
2. **Model Architecture**:
    - Build an encoder-decoder architecture using Keras.
    - The encoder takes the input English sequence and generates a fixed-size context vector that captures the meaning of the input sentence.
    - The decoder takes the context vector and generates the output Marathi sequence one token at a time.
    - Both the encoder and decoder are recurrent neural networks (RNNs), typically implemented using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells.
    - The final output of the decoder is a probability distribution over the Marathi vocabulary, indicating the likelihood of each word in the output sequence.
3. **Model Training**:
    - Train the Seq2Seq model on the parallel corpus using teacher forcing.

- Teacher forcing is a technique where, during training, the decoder is fed the correct target tokens from the previous time step as input, rather than its own predictions.
- The loss function used during training is typically categorical cross-entropy, which measures the difference between the predicted and actual probability distributions over the vocabulary.

2. **Inference**:
    - During inference (i.e., translation of new sentences), the encoder-decoder model is used to generate the Marathi translation of the input English sentence.
    - The encoder processes the input sentence to generate the context vector.
    - The decoder starts with a special "start of sequence" token and iteratively generates the output tokens one by one until it generates an "end of sequence" token or reaches a maximum length.
    - At each step, the decoder's output token is fed back into the decoder as input for the next step, along with the current context vector.
3. **Model Evaluation**:
    - Evaluate the performance of the trained model using metrics such as BLEU score, which measures the similarity between the predicted and reference translations.
    - Additionally, perform qualitative analysis by inspecting sample translations to assess the model's fluency and accuracy.

Code snippet

```
!pip install pandas
!pip install keras
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.15.0)
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
import string
from string import digits
import re
from sklearn.utils import shuffle
from keras.preprocessing.sequence import pad_sequences
from keras.layers import LSTM, Input, Dense,Embedding
from keras.models import Model,load_model
from keras.utils import plot_model
from keras.preprocessing.text import one_hot
from keras.preprocessing.text import Tokenizer
from keras.models import model_from_json
import pickle as pkl
import numpy as np
```

```python
with open('mar.txt','r') as f:
  data = f.read()
# len(data)
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help

Comment Share

+ Code + Text

Connect

```
# we need to clean the data
uncleaned_data_list = data.split('\n')
print(len(uncleaned_data_list))

uncleaned_data_list = uncleaned_data_list[:38695]
print(len(uncleaned_data_list))

english_word = []
marathi_word = []
# cleaned_data_list = []
for word in uncleaned_data_list:
  english_word.append(word.split('\t')[:-1][0])
  marathi_word.append(word.split('\t')[:-1][1])

print(len(english_word), len(marathi_word))
```

```
46996
38695
38695 38695
```

```
language_data = pd.DataFrame(columns=['English','Marathi'])
language_data['English'] = english_word
language_data['Marathi'] = marathi_word

# saving to csv
language_data.to_csv('language_data.csv', index=False)

# loading data from csv
language_data = pd.read_csv('language_data.csv')
```

```
language_data.head()
```

English **Marathi**

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
language_data.head()
```

	English	Marathi
0	Go.	जा.
1	Run!	पळ!
2	Run!	धाव!
3	Run!	पळा!
4	Run!	धावा!

```
language_data.tail()
```

	English	Marathi
38690	I don't see your name on the list.	मला तुमचं नाव यादीत दिसत नाहीये.
38691	I don't see your name on the list.	मला तुमचं नाव यादीत दिसत नाही.
38692	I don't think that was your fault.	तुझी चूक होती असं मला वाटत नाही.
38693	I don't think that was your fault.	तुमची चूक होती असं मला वाटत नाही.
38694	I don't understand English at all.	मला इंग्रजी अजिबात समजत नाही.

```
#text preprocessing which includes some basic things like lowercase the text, removing punctuation, removing digits, and whitespace.
english_text = language_data['English'].values
marathi_text = language_data['Marathi'].values

english_text[0], marathi_text[0]

#lowercasing the setences
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect

```python
#lowercasing the setences
english_text_ = [x.lower() for x in english_text]
marathi_text_ = [x.lower() for x in marathi_text]

# Text preprocessing
english_text_ = [re.sub("'",'',x) for x in english_text_]
marathi_text_ = [re.sub("'",'',x) for x in marathi_text_]

# remove puntuation
def remove_punc(text_list):
  table = str.maketrans('', '', string.punctuation)
  removed_punc_text = []
  for sent in text_list:
    sentance = [w.translate(table) for w in sent.split(' ')]
    removed_punc_text.append(' '.join(sentance))
  return removed_punc_text
english_text_ = remove_punc(english_text_)
marathi_text_ = remove_punc(marathi_text_)

# removing the digits from english sentances
remove_digits = str.maketrans('', '', digits)
removed_digits_text = []
for sent in english_text_:
  sentance = [w.translate(remove_digits) for w in sent.split(' ')]
  removed_digits_text.append(' '.join(sentance))
english_text_ = removed_digits_text

# removing the digits from the marathi sentances
marathi_text_ = [re.sub("[२३०८१५७९४६]","",x) for x in marathi_text_]
marathi_text_ = [re.sub("[\u200d]","",x) for x in marathi_text_]

# removing the stating and ending whitespaces
english_text_ = [x.strip() for x in english_text_]
marathi_text_ = [x.strip() for x in marathi_text_]
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Connect

```
# Putting the start and end words in the marathi sentances
marathi_text_ = ["start " + x + " end" for x in marathi_text_]
```

```
#Splitting our dataset

X = english_text_
Y = marathi_text_

X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size = 0.1)

print(len(X_train))

print(len(y_train))

print(len(X_test))

print(len(y_test))
```

```
34825
34825
3870
3870
```

```
def generator_batch(X=X_train, Y=y_train, batch_size=128, tokenizer_input=None, tokenizer_target=None):
    while True:
        for j in range(0, len(X), batch_size):
            encoder_data_input = np.zeros((batch_size, max_length_english), dtype='float32')
            decoder_data_input = np.zeros((batch_size, max_length_marathi), dtype='float32')
            decoder_target_input = np.zeros((batch_size, max_length_marathi, vocab_size_target), dtype='float32')
            for i, (input_text, target_text) in enumerate(zip(X[j:j + batch_size], Y[j:j + batch_size])):
                for t, word in enumerate(input_text.split()):
                    encoder_data_input[i, t] = tokenizer_input.word_index[word]
                for t, word in enumerate(target_text.split()):
                    decoder_data_input[i, t] = tokenizer_target.word_index[word]
```

```
                decoder_data_input[i, t] = tokenizer_target.word_index[word]
                if t > 0:
                    decoder_target_input[i, t - 1, tokenizer_target.word_index[word]] = 1
        yield ([encoder_data_input, decoder_data_input], decoder_target_input)
```

```
#These tokenizers are essential for converting text data into numerical sequences, which are then used as inputs to your model.
from keras.models import Model
from keras.layers import Input, LSTM, Dense, Embedding
import numpy as np

latent_dim = 50

# Initialize and fit tokenizers
input_tokenizer = Tokenizer()
target_tokenizer = Tokenizer()
input_tokenizer.fit_on_texts(X_train)
target_tokenizer.fit_on_texts(y_train)

# Compute vocabulary sizes
vocab_size_input = len(input_tokenizer.word_index) + 1
vocab_size_target = len(target_tokenizer.word_index) + 1

# Compute maximum sequence lengths
max_length_english = max(len(sequence.split()) for sequence in X_train)
max_length_marathi = max(len(sequence.split()) for sequence in y_train)

# Define the input sequences
encoder_inputs = Input(shape=(None,), name="encoder_inputs")
decoder_inputs = Input(shape=(None,), name="decoder_inputs")

# Define the encoder
emb_layer_encoder = Embedding(vocab_size_input, latent_dim, mask_zero=True)(encoder_inputs)
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(emb_layer_encoder)
encoder_states = [state_h, state_c]
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
# Define the decoder
emb_layer_decoder = Embedding(vocab_size_target, latent_dim, mask_zero=True)(decoder_inputs)
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(emb_layer_decoder, initial_state=encoder_states)
decoder_dense = Dense(vocab_size_target, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# Define the model
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

# Define generator parameters
train_samples = len(X_train)
batch_size = 128
epochs = 50

# Fit the model
model.fit_generator(generator=generator_batch(X_train, y_train, batch_size=batch_size,
                                               tokenizer_input=input_tokenizer,
                                               tokenizer_target=target_tokenizer),
                    steps_per_epoch=train_samples // batch_size,
                    epochs=epochs)
```

```
<ipython-input-42-67fc27e5a691>:50: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supp
  model.fit_generator(generator=generator_batch(X_train, y_train, batch_size=batch_size,
Epoch 1/50
272/272 [==============================] - 188s 659ms/step - loss: 5.7785 - accuracy: 0.1585
Epoch 2/50
272/272 [==============================] - 168s 618ms/step - loss: 4.8183 - accuracy: 0.1814
Epoch 3/50
272/272 [==============================] - 166s 611ms/step - loss: 4.5942 - accuracy: 0.1869
Epoch 4/50
272/272 [==============================] - 180s 661ms/step - loss: 4.4907 - accuracy: 0.1929
Epoch 5/50
272/272 [==============================] - 168s 617ms/step - loss: 4.4087 - accuracy: 0.1992
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect

```
Epoch 23/50
272/272 [==============================] - 164s 605ms/step - loss: 3.2717 - accuracy: 0.3381
Epoch 24/50
272/272 [==============================] - 166s 609ms/step - loss: 3.2280 - accuracy: 0.3432
Epoch 25/50
272/272 [==============================] - 167s 613ms/step - loss: 3.1867 - accuracy: 0.3480
Epoch 26/50
272/272 [==============================] - 181s 667ms/step - loss: 3.1471 - accuracy: 0.3525
Epoch 27/50
272/272 [==============================] - 171s 627ms/step - loss: 3.1091 - accuracy: 0.3568
Epoch 28/50
272/272 [==============================] - 170s 625ms/step - loss: 3.0711 - accuracy: 0.3613
Epoch 29/50
272/272 [==============================] - 181s 668ms/step - loss: 3.0345 - accuracy: 0.3652
Epoch 30/50
272/272 [==============================] - 166s 610ms/step - loss: 2.9997 - accuracy: 0.3690
Epoch 31/50
272/272 [==============================] - 166s 611ms/step - loss: 2.9654 - accuracy: 0.3726
Epoch 32/50
```

```python
model_json = model.to_json()
with open("model_2.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model_weight_5.h5")
print("Saved model to disk")

# loading the model architecture and asigning the weights
json_file = open('model_2.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model_loaded = model_from_json(loaded_model_json)
# load weights into new model
model_loaded.load_weights("model_weight_5.h5")
```

```
Saved model to disk
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
model_loaded.load_weights("model_weight_5.h5")
```

```
Saved model to disk
```

```
latent_dim = 50
#inference encoder
encoder_inputs_inf = model_loaded.input[0] #Trained encoder input layer
encoder_outputs_inf, inf_state_h, inf_state_c = model_loaded.layers[4].output # retoring the encoder lstm output and states
encoder_inf_states = [inf_state_h,inf_state_c]
encoder_model = Model(encoder_inputs_inf,encoder_inf_states)


#inference decoder
# The following tensor will store the state of the previous timestep in the "starting the encoder final time step"
decoder_state_h_input = Input(shape=(latent_dim,)) #becase during training we have set the lstm unit to be of 50
decoder_state_c_input = Input(shape=(latent_dim,))
decoder_state_input = [decoder_state_h_input,decoder_state_c_input]

# # inference decoder input
decoder_input_inf = model_loaded.input[1] #Trained decoder input layer
# decoder_input_inf._name='decoder_input'
decoder_emb_inf = model_loaded.layers[3](decoder_input_inf)
decoder_lstm_inf = model_loaded.layers[5]
decoder_output_inf, decoder_state_h_inf, decoder_state_c_inf = decoder_lstm_inf(decoder_emb_inf, initial_state =decoder_state_input)
decoder_state_inf = [decoder_state_h_inf,decoder_state_c_inf]
#inference dense layer
dense_inf = model_loaded.layers[6]
decoder_output_final = dense_inf(decoder_output_inf)# A dense softmax layer to generate prob dist. over the target vocabulary

decoder_model = Model([decoder_input_inf]+decoder_state_input,[decoder_output_final]+decoder_state_inf)
```

```
# Code to predct the input sentences translation
reverse_word_map_target = {v: k for k, v in target_tokenizer.word_index.items()}
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect

```python
# Code to predct the input sentences translation
reverse_word_map_target = {v: k for k, v in target_tokenizer.word_index.items()}
def decode_seq(input_seq):
  # print("input_seq=>",input_seq)
  state_values_encoder = encoder_model.predict(input_seq)
  # intialize the target seq with start tag
  target_seq = np.zeros((1,1))
  target_seq[0, 0] = target_tokenizer.word_index['start']
  # print("target_seq:=>",target_seq)
  stop_condition = False
  decoder_sentance = ''
  # print("Beforee the while loop")
  while not stop_condition:
    sample_word , decoder_h,decoder_c= decoder_model.predict([target_seq] + state_values_encoder)
    # print("sample_word: =>",sample_word)
    sample_word_index = np.argmax(sample_word[0,-1,:])
    # print("sample_word_index: ",sample_word_index)
    decoder_word = reverse_word_map_target[sample_word_index]
    decoder_sentance += ' '+ decoder_word
    # print("decoded word:=>",decoder_word)
    # print(len(decoder_sentance))
    # print("len(decoder_sentance) > 70: ",len(decoder_sentance) > 70)
    # print('decoder_word == "end"',decoder_word == 'end')
    # print(decoder_word == 'end' or len(decoder_sentance) > 70)
    # stop condition for the while loop
    if (decoder_word == 'end' or
        len(decoder_sentance) > 70):
        stop_condition = True
        # print("from if condition")
    # target_seq = np.zeros((1,1))
    target_seq[0, 0] = sample_word_index
    # print(target_seq)
    state_values_encoder = [decoder_h,decoder_c]
  return decoder_sentance
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```python
for i in range(30):
  sentance = X_test[i]
  original_target = y_test[i]
  input_seq = input_tokenizer.texts_to_sequences([sentance])
  pad_sequence = pad_sequences(input_seq, maxlen= 30,padding='post')
  # print('input_sequence =>',input_seq)
  # print("pad_seq=>",pad_sequence)
  predicted_target = decode_seq(pad_sequence)
  print("Test sentance: ",i+1)
  print("sentance: ",sentance)
  print("origianl translate:",original_target[6:-4])
  print("predicted Translate:",predicted_target[:-4])
  print("=="*50)
```

```
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 37ms/step
Test sentance:  1
sentance:  i must study
origianl translate: मला अभ्यास करायला हवा
predicted Translate:  मी अभ्यास करायला केलं
====================================================================================================
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 51ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 44ms/step
Test sentance:  2
sentance:  youve come too early
origianl translate: खूपच लवकर आला आहेस
predicted Translate:  तुम्ही खूपच वाजता सुरु केलं
====================================================================================================
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connect

```
Test sentance:  3
sentance:  were historians
origianl translate: आम्ही इतिहासकार आहोत
predicted Translate:  आपण शांत आहे
====================================================================================================
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 38ms/step
Test sentance:  4
sentance:  a thief believes everybody steals
origianl translate: चोर मानतो की सगळेच चोरतात
predicted Translate:  हे घरी ते घरी का
====================================================================================================
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 57ms/step
Test sentance:  5
sentance:  wheres tom beats me
origianl translate: टॉम कुठे आहे काय माहीत
predicted Translate:  टॉम कुठे आहे
====================================================================================================
1/1 [==============================] - 0s 102ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 98ms/step
1/1 [==============================] - 0s 145ms/step
1/1 [==============================] - 0s 133ms/step
Test sentance:  6
sentance:  whose fault is it
origianl translate: ती कोणाची चूक आहे
predicted Translate:  चूक कोण आहे
====================================================================================================
1/1 [==============================] - 0s 109ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 113ms/step
1/1 [==============================] - 0s 37ms/step
```

colab.research.google.com

NLP_PROJECT_MT_15_28.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
1/1 [==============================] - 0s 37ms/step
Test sentance:  7
sentance:  dont look
origianl translate: बघू नकोस
predicted Translate:  विसरू नकोस
====================================================================================================
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 36ms/step
Test sentance:  8
sentance:  i listened to his story
origianl translate: मी त्याची गोष्ट ऐकली
predicted Translate:  मी आपली मदत करतो
====================================================================================================
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 84ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 58ms/step
Test sentance:  9
sentance:  the gasoline tank was underneath
origianl translate: पेट्रोलची टाकी खाली होती
predicted Translate:  सफरचंद या
====================================================================================================
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 38ms/step
Test sentance:  10
sentance:  i dont know them at all
origianl translate: मी त्यांना अजिबात ओळखत नाही
predicted Translate:  मी आता खोटं करायला केलं
====================================================================================================
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 74ms/step
```

NLP_PROJECT_MT_15_28.ipynb


```
1/1 [==============================] - 0s 37ms/step
Test sentance:  7
sentance:  dont look
origianl translate: बघू नकोस
predicted Translate:  विसरू नकोस
================================================================================
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 36ms/step
Test sentance:  8
sentance:  i listened to his story
origianl translate: मी त्याची गोष्ट ऐकली
predicted Translate:  मी आपली मदत करतो
================================================================================
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 84ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 58ms/step
Test sentance:  9
sentance:  the gasoline tank was underneath
origianl translate: पेट्रोलची टाकी खाली होती
predicted Translate:  सफरचंद या
================================================================================
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 38ms/step
Test sentance:  10
sentance:  i dont know them at all
origianl translate: मी त्यांना अजिबात ओळखत नाही
predicted Translate:  मी आता खोटं करायला केलं
================================================================================
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 0s 74ms/step
```

Sequence-to-sequence (Seq2Seq) models for machine translation have several advantages and disadvantages compared to other approaches. Let's compare Seq2Seq models with two other common machine translation approaches: Statistical Machine Translation (SMT) and Transformer-based models.

1. **Statistical Machine Translation (SMT)**:
    - **Advantages**:
        - SMT models are based on probabilistic models and linguistic rules, making them interpretable and easy to understand.
        - They can handle rare or unseen words better than neural network-based models.
    - **Disadvantages**:
        - SMT models require hand-crafted features and extensive engineering of linguistic rules, which can be labor-intensive and time-consuming.
        - They struggle with capturing long-range dependencies and contextual information effectively.
        - Performance tends to degrade with the complexity of languages and the size of the vocabulary.

2. Transformer-based Models (e.g., BERT, GPT):

- **Advantages**:
    - Transformer-based models excel at capturing long-range dependencies and contextual information through self-attention mechanisms.
    - They can handle large vocabularies and diverse language pairs effectively.
    - Pre-trained transformer models (e.g., BERT, GPT) can be fine-tuned for specific translation tasks, leveraging transfer learning.
- **Disadvantages**:
    - They require large amounts of computational resources and data for training, making them less accessible for smaller organizations or research projects.
    - Transformer models can be less interpretable compared to traditional SMT models, making it harder to debug or understand model behavior.
    - Fine-tuning pre-trained transformer models for machine translation tasks may require substantial computational resources and expertise.

3. Seq2Seq Models:

- **Advantages**:
  - Seq2Seq models are simpler to implement and train compared to transformer-based models.
  - They are effective at capturing the overall structure of the input sequence and generating coherent output sequences.
  - Seq2Seq models can be trained end-to-end, allowing for joint optimization of both the encoder and decoder.
- **Disadvantages**:
  - They struggle with handling long input sequences and capturing fine-grained contextual information.
  - Performance may degrade when translating between languages with different word orders or morphological structures.
  - Seq2Seq models may suffer from exposure bias and generate generic or repetitive translations, especially for rare or complex sentences.

In summary, Seq2Seq models offer a balance between simplicity and effectiveness, making them suitable for various machine translation tasks. However, transformer-based models have shown superior performance in recent years, especially for large-scale and complex translation tasks, albeit at the cost of increased computational requirements. SMT models remain relevant for tasks where interpretability and linguistic rules play a crucial role, despite their limitations in capturing complex language patterns.

Thankyou