Installing Modules

```
!pip install pytorch_lightning torchmetrics tableprint spacy==3
!python -m spacy download en_core_web_sm
!python -m spacy download de_core_news_sm
```

```
Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=3.7.4; python_version < "
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyth
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5; python_version < "3.8" in /usr/lo
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /usr/local/lib/p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python
Installing collected packages: en-core-web-sm
  Found existing installation: en-core-web-sm 2.2.5
    Uninstalling en-core-web-sm-2.2.5:
      Successfully uninstalled en-core-web-sm-2.2.5
Successfully installed en-core-web-sm-3.0.0
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
2021-07-01 04:30:25.973252: I tensorflow/stream_executor/platform/default/d
Collecting de-core-news-sm==3.0.0
  Downloading https://github.com/explosion/spacy-models/releases/download/d
     |████████████████████████████████| 19.3MB 1.4MB/s
Requirement already satisfied: spacy<3.1.0,>=3.0.0 in /usr/local/lib/python
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-

Requirement already satisfied: srsly<3.0.0,>=2.4.0 in /usr/local/lib/python
Requirement already satisfied: thinc<8.1.0,>=8.0.0 in /usr/local/lib/python
Requirement already satisfied: catalogue<2.1.0,>=2.0.1 in /usr/local/lib/py
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.0 in /usr/local/lib
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/pytho
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: importlib-metadata>=0.20; python_version < "
Requirement already satisfied: pydantic<1.8.0,>=1.7.1 in /usr/local/lib/pyt
Requirement already satisfied: typing-extensions>=3.7.4; python_version < "
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/py
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyth
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python
Requirement already satisfied: pathy in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typer<0.4.0,>=0.3.0 in /usr/local/lib/python
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7
Requirement already satisfied: zipp>=0.5; python_version < "3.8" in /usr/lo
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /usr/local/lib/py
Requirement already satisfied: click<7.2.0,>=7.1.1 in /usr/local/lib/python3
Installing collected packages: de-core-news-sm
Successfully installed de-core-news-sm-3.0.0
✔ Download and installation successful
```

Imports

```
# Import Libraries
import random
from typing import Iterable, List, Tuple
import pandas as pd
import sys, os, pickle
import numpy as np
import math
import matplotlib.pyplot  as plt
import spacy

# PyTorch related
import torch, torchtext
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torch import Tensor
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator
from torchtext.datasets import Multi30k
from torch.nn.utils.rnn import pad_sequence
from torch.utils.data import DataLoader

# My Custom Code
import pytorch_lightning as pl
import torchmetrics
from pytorch_lightning.loggers import CSVLogger
from pytorch_lightning.callbacks import ModelCheckpoint
import tableprint as tp
```

```
/usr/local/lib/python3.7/dist-packages/pytorch_lightning/metrics/__init__.py:
  "`pytorch_lightning.metrics.*` module has been renamed to `torchmetrics.*`
```

```
# Manual Seed
SEED = 1234

random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
torch.backends.cudnn.deterministic = True


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

Language Definitions

```
SRC_LANGUAGE = 'de'
TGT_LANGUAGE = 'en'

# Place-holders
token_transform = {}
vocab_transform = {}
```

Tokenizers

```
token_transform[SRC_LANGUAGE]  = get_tokenizer('spacy', language='de_core_news_sm'
token_transform[TGT_LANGUAGE] = get_tokenizer('spacy', language='en_core_web_sm')
```

Yield Function

This yields the tokens for the texts and will be used to build the vocab

```
def yield_tokens(data_iter: Iterable, language: str) -> List[str]:
    language_index = {SRC_LANGUAGE: 0, TGT_LANGUAGE: 1}

    for data_sample in data_iter:
        yield token_transform[language](data_sample[language_index[language]])
```

Special Tokens

```
# Define special symbols and indices
UNK_IDX, PAD_IDX, BOS_IDX, EOS_IDX = 0, 1, 2, 3
# Make sure the tokens are in order of their indices to properly insert them in vo
special_symbols = ['<unk>', '<pad>', '<bos>', '<eos>']
```

Build the vocab here

```
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
  # Training data Iterator
  train_iter = Multi30k(split='train', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
  # Create torchtext's Vocab object
  vocab_transform[ln] = build_vocab_from_iterator(yield_tokens(train_iter, ln),
                                                  min_freq=1,
                                                    specials=special_symbols,
                                                    special_first=True)
```

```
training.tar.gz: 100%|██████████| 1.21M/1.21M [00:01<00:00, 1.08MB/s]
```

Setting the default index as the token

```
# Set UNK_IDX as the default index. This index is returned when the token is not f
# If not set, it throws RuntimeError when the queried token is not found in the Vo
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
  vocab_transform[ln].set_default_index(UNK_IDX)
```

```
len(vocab_transform['de'])
```

```
19215
```

```
len(vocab_transform['en'])
```

```
10838
```

Collator

```
# helper function to club together sequential operations
def sequential_transforms(*transforms):
    def func(txt_input):
        for transform in transforms:
            txt_input = transform(txt_input)
        return txt_input
    return func

# function to add BOS/EOS and create tensor for input sequence indices
def tensor_transform(token_ids: List[int]):
    return torch.cat((torch.tensor([BOS_IDX]),
                      torch.tensor(token_ids),
                      torch.tensor([EOS_IDX])))

# src and tgt language text transforms to convert raw strings into tensors indices
text_transform = {}
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
    text_transform[ln] = sequential_transforms(token_transform[ln], #Tokenization
                                               vocab_transform[ln], #Numericalizat
                                               tensor_transform) # Add BOS/EOS and
```

```
# function to collate data samples into batch tesors
def collate_fn(batch):
    src_batch, tgt_batch = [], []
    for src_sample, tgt_sample in batch:
        src_batch.append(text_transform[SRC_LANGUAGE](src_sample.rstrip("\n")))
        tgt_batch.append(text_transform[TGT_LANGUAGE](tgt_sample.rstrip("\n")))

    src_batch = pad_sequence(src_batch, padding_value=PAD_IDX)
    tgt_batch = pad_sequence(tgt_batch, padding_value=PAD_IDX)
    return src_batch, tgt_batch
```

DataLoader

```
BATCH_SIZE = 32
train_iter = Multi30k(split='train', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
train_loader = DataLoader(train_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn

val_iter = Multi30k(split='valid', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
val_loader = DataLoader(val_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn, nu

test_iter = Multi30k(split='test', language_pair=(SRC_LANGUAGE, TGT_LANGUAGE))
test_loader = DataLoader(test_iter, batch_size=BATCH_SIZE, collate_fn=collate_fn,
```

```
validation.tar.gz: 100%|██████████| 46.3k/46.3k [00:00<00:00, 168kB/s]
mmt16_task1_test.tar.gz: 100%|██████████| 43.9k/43.9k [00:00<00:00, 158kB/s]
```

Model

Boilerplate Code for PyTorch Lightning

```
class TL(pl.LightningModule):
    def __init__(self):
        super(TL, self).__init__()

        self.train_acc =  torch.tensor(0.)
        self.avg_train_loss = torch.tensor(0.)
        self.table_context = None


    def training_step(self, batch, batch_idx):
        src, trg = batch
        output = self(src, trg)
        output_dim = output.shape[-1]
        output = output[1:].view(-1, output_dim)
        trg = trg[1:].view(-1)
        loss_train = self.loss(output, trg)
        return loss_train

    def validation_step(self, batch, batch_idx):
        src, trg = batch
        output = self(src, trg, 0)
        output_dim = output.shape[-1]
        output = output[1:].view(-1, output_dim)
        trg = trg[1:].view(-1)
        loss_valid = self.loss(output, trg)
        return {"loss": loss_valid}

    def training_epoch_end(self, outputs):
        self.avg_train_loss = torch.stack([x['loss'] for x in outputs]).mean()

    def validation_epoch_end(self, outputs):
        if trainer.sanity_checking:
          print('sanity check')
          return
        avg_valid_loss = torch.stack([x['loss'] for x in outputs]).mean()
        metrics = {'epoch': self.current_epoch+1, 'Train PPL': math.exp(self.avg_t
        if self.table_context is None:
          self.table_context = tp.TableContext(headers=['epoch', 'Train PPL', 'Tra
          self.table_context.__enter__()
        self.table_context([self.current_epoch+1, math.exp(self.avg_train_loss.ite
        self.logger.log_metrics(metrics)
        if self.current_epoch == self.trainer.max_epochs - 1:
          self.validation_end(outputs)

    def validation_end(self, outputs):
        self.table_context.__exit__()
```

Encoder

```
class Encoder(pl.LightningModule):
    def __init__(self, input_dim, emb_dim, enc_hid_dim, dec_hid_dim, dropout):
        super().__init__()

        self.hid_dim = enc_hid_dim

        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.GRU(emb_dim, enc_hid_dim, bidirectional = True)
        self.fc = nn.Linear(enc_hid_dim * 2, dec_hid_dim)
        self.dropout = nn.Dropout(dropout)

    def forward(self, src):
        embedded = self.dropout(self.embedding(src))
        output, hidden = self.rnn(embedded)
        hidden = torch.tanh(self.fc(torch.cat((hidden[-2,:,:], hidden[-1,:,:]), di

        return output, hidden
```

Attention

```
class Attention(pl.LightningModule):
    def __init__(self, enc_hid_dim, dec_hid_dim):
        super().__init__()

        self.attn = nn.Linear((enc_hid_dim * 2) + dec_hid_dim, dec_hid_dim)
        self.v = nn.Linear(dec_hid_dim, 1, bias = False)


    def forward(self, hidden, encoder_outputs):
        batch_size = encoder_outputs.shape[1]
        src_len = encoder_outputs.shape[0]
        hidden = hidden.unsqueeze(1).repeat(1, src_len, 1)
        encoder_outputs = encoder_outputs.permute(1, 0, 2)
        energy = torch.tanh(self.attn(torch.cat((hidden, encoder_outputs), dim = 2

        attention = self.v(energy).squeeze(2)
        return F.softmax(attention, dim=1)
```

Decoder

```
class Decoder(pl.LightningModule):
    def __init__(self, output_dim, emb_dim, enc_hid_dim, dec_hid_dim, dropout, att
        super().__init__()

        self.hid_dim = dec_hid_dim
        self.output_dim = output_dim
        self.attention = attention



        self.embedding = nn.Embedding(output_dim, emb_dim)

        self.rnn = nn.GRU((enc_hid_dim * 2) + emb_dim, dec_hid_dim)

        self.fc_out = nn.Linear((enc_hid_dim * 2) + dec_hid_dim + emb_dim, output_

        self.dropout = nn.Dropout(dropout)

    def forward(self, input, hidden, encoder_outputs):
        input = input.unsqueeze(0)
        embedded = self.dropout(self.embedding(input))

        a = self.attention(hidden, encoder_outputs)

        a = a.unsqueeze(1)

        encoder_outputs = encoder_outputs.permute(1, 0, 2)

        weighted = torch.bmm(a, encoder_outputs)

        weighted = weighted.permute(1, 0, 2)

        rnn_input = torch.cat((embedded, weighted), dim = 2)

        output, hidden = self.rnn(rnn_input, hidden.unsqueeze(0))

        assert (output == hidden).all()
        assert (output == hidden).all()

        embedded = embedded.squeeze(0)
        output = output.squeeze(0)
        weighted = weighted.squeeze(0)

        prediction = self.fc_out(torch.cat((output, weighted, embedded), dim = 1))

        return prediction, hidden.squeeze(0)
```

Seq2Seq Model

```
# Define the model

class Seq2Seq(TL):
    def __init__(self, encoder, decoder, device):
        super(Seq2Seq, self).__init__()

        self.loss = nn.CrossEntropyLoss(ignore_index=PAD_IDX)
        self.lr = 1e-3

        self.encoder = encoder
        self.decoder = decoder
        # self.device = device # Doesn't work in PyTorchLightning since it is alre


    def forward(self, src, trg, teacher_forcing_ratio = 0.5):

        batch_size = trg.shape[1]
        trg_len = trg.shape[0]
        trg_vocab_size = self.decoder.output_dim

        outputs = torch.zeros(trg_len, batch_size, trg_vocab_size).to(self.device)

        encoder_outputs, hidden = self.encoder(src)

        input = trg[0,:]

        for t in range(1, trg_len):

            output, hidden = self.decoder(input, hidden, encoder_outputs)

            outputs[t] = output

            teacher_force = random.random() < teacher_forcing_ratio

            top1 = output.argmax(1)

            input = trg[t] if teacher_force else top1

        return outputs

    def configure_optimizers(self):
        optim = torch.optim.Adam(self.parameters())
        optim = torch.optim.Adam(self.parameters())
        return optim
```

Model Initialization and Summary

```
INPUT_DIM = len(vocab_transform[SRC_LANGUAGE])
OUTPUT_DIM = len(vocab_transform[TGT_LANGUAGE])

ENC_EMB_DIM = 256
DEC_EMB_DIM = 256
ENC_HID_DIM = 512
DEC_HID_DIM = 512
ENC_DROPOUT = 0.5
DEC_DROPOUT = 0.5


attn = Attention(ENC_HID_DIM, DEC_HID_DIM)
enc = Encoder(INPUT_DIM, ENC_EMB_DIM, ENC_HID_DIM, DEC_HID_DIM, ENC_DROPOUT)
dec = Decoder(OUTPUT_DIM, DEC_EMB_DIM, ENC_HID_DIM, DEC_HID_DIM, DEC_DROPOUT, attn

model = Seq2Seq(enc, dec, device).to(device)
```

Model Checkpoint

```
checkpoint_callback = ModelCheckpoint(
    monitor='val_loss',
    dirpath='/content',
    filename='sst-{epoch:02d}-{val_loss:.2f}',
    mode='min'
)
```

```
!rm -rf csv_logs
csvlogger = CSVLogger('csv_logs', name='END2_Assign_8', version=0)
trainer = pl.Trainer(max_epochs=10, num_sanity_val_steps=1, logger=csvlogger, gpus
trainer.fit(model, train_dataloader=train_loader, val_dataloaders=val_loader)
checkpoint_callback.best_model_path
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name    | Type             | Params
---------------------------------------------
0 | loss    | CrossEntropyLoss | 0
1 | encoder | Encoder          | 7.8 M
2 | decoder | Decoder          | 25.8 M
---------------------------------------------
33.6 M    Trainable params
0         Non-trainable params
33.6 M    Total params
134.238   Total estimated model params size (MB)
```

Validation sanity check: 0% 0/1 [18:30<?, ?it/s]

```
/usr/local/lib/python3.7/dist-packages/pytorch_lightning/utilities/data.py:42
  'Your `IterableDataset` has `__len__` defined.'
sanity check
```

Epoch 9: 100% 939/939 [02:26<00:00, 6.42it/s, loss=1.84, v_num=0]

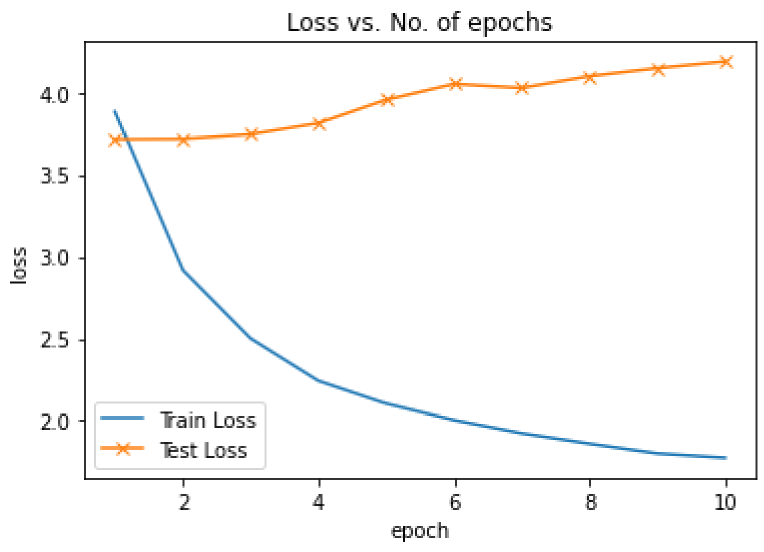
```
┌──────────────┬──────────────┬──────────────┬──────────────┬──────────────┐
│        epoch │    Train PPL │   Train Loss │    Valid PPL │   Valid Loss │
├──────────────┼──────────────┼──────────────┼──────────────┼──────────────┤
│            1 │       48.576 │       3.8831 │       41.079 │       3.7155 │
```

Training Log

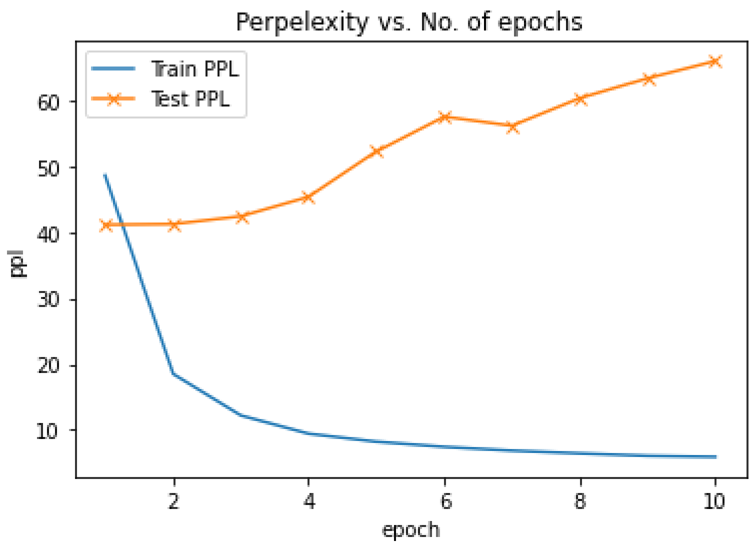
```
│            5 │       8.2057 │       2.1048 │       52.213 │       3.9553 │
```

```
root='./csv_logs/' + 'END2_Assign_8' + '/'
dirlist = [ item for item in os.listdir(root) if os.path.isdir(os.path.join(root, i
metricfile = root + dirlist[-1:][0] + '/metrics.csv'
metrics = pd.read_csv(metricfile)


plt.plot(metrics['epoch'], metrics['Train Loss'], label="Train Loss")
plt.plot(metrics['epoch'], metrics['Valid Loss'], '-x', label="Test Loss")
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend()
plt.title('Loss vs. No. of epochs');
```



```
plt.plot(metrics['epoch'], metrics['Train PPL'], label="Train PPL")
plt.plot(metrics['epoch'], metrics['Valid PPL'], '-x', label="Test PPL")
plt.xlabel('epoch')
plt.ylabel('ppl')
plt.legend()
plt.title('Perpelexity vs. No. of epochs');
```



Inference on Random Samples from Test Data

```
model.to(device)
model.eval()
```

```
    Seq2Seq(
      (loss): CrossEntropyLoss()
      (encoder): Encoder(
        (embedding): Embedding(19215, 256)
        (rnn): GRU(256, 512, bidirectional=True)
        (fc): Linear(in_features=1024, out_features=512, bias=True)
        (dropout): Dropout(p=0.5, inplace=False)
      )
      (decoder): Decoder(
        (attention): Attention(
          (attn): Linear(in_features=1536, out_features=512, bias=True)
          (v): Linear(in_features=512, out_features=1, bias=False)
        )
        (embedding): Embedding(10838, 256)
        (rnn): GRU(1280, 512)
        (fc_out): Linear(in_features=1792, out_features=10838, bias=True)
        (dropout): Dropout(p=0.5, inplace=False)
      )
    )
```

```
for i in np.random.randint(0,32, 10):
  src_sent_i = next(iter(test_loader))[0][:,i]
  trg_sent_i = next(iter(test_loader))[1][:,i]
  stop_ind_src = (src_sent_i==3).nonzero()[0].item() # stop when <eos> token is fo
  stop_ind_trg = (trg_sent_i==3).nonzero()[0].item() # stop when <eos> token is fo
  src_sent_tok = [vocab_transform['de'].lookup_token(word_i) for word_i in src_sen
```

```
    src_sent_tok = [vocab_transform['de'].lookup_token(word_i) for word_i in src_sen
    trg_sent_tok = [vocab_transform['en'].lookup_token(word_i) for word_i in trg_sent
    src_sent = " ".join(src_sent_tok[1:]) # skip the initial <bos> token
    trg_sent = " ".join(trg_sent_tok[1:]) # skip the initial <bos> token
    src_sent_tensor = src_sent_i.clone().detach().unsqueeze(1).to(device)
    trg_sent_tensor = trg_sent_i.clone().detach().unsqueeze(1).to(device)
    with torch.no_grad():
        output = model(src_sent_tensor, trg_sent_tensor, 1)
        out = output.squeeze(1)
        out = torch.argmax(out,dim=1)
        stop_ind_pred = (out==3).nonzero()[0].item() # stop when <eos> token is fou
        trans = []
        pred_sent_tok = [vocab_transform['en'].lookup_token(word_i) for word_i in o
        pred_sent = " ".join(pred_sent_tok[1:stop_ind_pred])
        start = "\033[1m"
        end = "\033[0;0m"
        print(f'{start}Source Sentence: {end}{src_sent}')
        print(f'{start}Target Sentence: {end}{trg_sent}')
        print(f'{start}Translated Sentence: {end}{pred_sent}')
        print()
```

```
Source Sentence: Ein Mädchen in einem Jeanskleid läuft über einen erhöhten Sc
Target Sentence: A girl in a jean dress is walking along a raised balance bea
Translated Sentence: A girl in a red is is a a a raised balance . .

Source Sentence: Zwei Männer tun so als seien sie Statuen , während Frauen ih
Target Sentence: Two men pretend to be <unk> while women look on .
Translated Sentence: Two men are as be as as as watch on .

Source Sentence: Eine Gruppe von Menschen steht vor einem Iglu .
Target Sentence: A group of people standing in front of an igloo .
Translated Sentence: A group of people standing in a of a igloo .

Source Sentence: Eine Teenagerin spielt bei einem Spiel Trompete auf dem Feld
Target Sentence: A teenager plays her trumpet on the field at a game .
Translated Sentence: A teenage plays in in in a game in a game .

Source Sentence: Eine Frau , die in einer Küche eine Schale mit Essen hält .
Target Sentence: A woman holding a bowl of food in a kitchen .
Translated Sentence: A woman cooking a bowl kitchen food in a kitchen .

Source Sentence: Leute , die vor einem Gebäude stehen .
Target Sentence: People standing outside of a building .
Translated Sentence: People standing outside in building building .

Source Sentence: Eine Frau verwendet eine Bohrmaschine während ein Mann sie f
Target Sentence: A woman uses a drill while another man takes her picture .
Translated Sentence: A woman uses a man while a man photographs photographs p

Source Sentence: Eine Frau in einem pinken Pulli und einer Schürze putzt eine
Target Sentence: A woman in a pink sweater and an apron , cleaning a table wi
Translated Sentence: A woman in a pink sweater and apron apron is is a table

Source Sentence: Ein Mädchen in einem Jeanskleid läuft über einen erhöhten Sc
Target Sentence: A girl in a jean dress is walking along a raised balance bea
Translated Sentence: A girl in a red is is a a a raised balance . .

Source Sentence: Eine Frau in einem <unk> Pulli und mit einer schwarzen Baseb
Target Sentence: A woman in a gray sweater and black baseball cap is standing
```

```
Translated Sentence: A woman in a black sweater and black black cap is standi
```

✓ 1s completed at 9:55 AM