```
# Install/update portalocker
!pip install -U portalocker

# Download English and French spaCy models
!python -m spacy download en_core_web_sm
!python -m spacy download fr_core_news_sm
```

```
Requirement already satisfied: portalocker in /usr/local/lib/python3.10/dist-packages (2.8.2)
2023-12-02 16:06:40.644001: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factor
2023-12-02 16:06:40.644054: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory
2023-12-02 16:06:40.644088: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS fact
2023-12-02 16:06:42.337799: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Collecting en-core-web-sm==3.6.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.6.0/en_core_web_sm-3.6.0-py3-none
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 99.9 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.7.0,>=3.6.0 in /usr/local/lib/python3.10/dist-packages (from en-core-web-sm==3.6.0) (
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->e
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0-
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-core-w
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-core-w
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0-
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy<3
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-core-web-sm==
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-core-web-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->en-core
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0-
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy<
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spac
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->sp
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.7.0,>=3.6.0-
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
2023-12-02 16:06:53.177213: E tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factor
2023-12-02 16:06:53.177272: E tensorflow/compiler/xla/stream_executor/cuda/cuda_fft.cc:609] Unable to register cuFFT factory
2023-12-02 16:06:53.177310: E tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:1518] Unable to register cuBLAS fact
2023-12-02 16:06:54.845860: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Collecting fr-core-news-sm==3.6.0
  Downloading https://github.com/explosion/spacy-models/releases/download/fr_core_news_sm-3.6.0/fr_core_news_sm-3.6.0-py3-no
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.3/16.3 MB 84.9 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.7.0,>=3.6.0 in /usr/local/lib/python3.10/dist-packages (from fr-core-news-sm==3.6.0)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->fr-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->f
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->fr-
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->fr
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->fr-
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0-
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->fr
```

```
from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator
from typing import Iterable, List
from torch.nn.utils.rnn import pad_sequence
from torch.utils.data import DataLoader, Dataset
from timeit import default_timer as timer
from torch.nn import Transformer
from torch import Tensor
from sklearn.model_selection import train_test_split
from tqdm.auto import tqdm
import torch.nn as nn
import torch
import torch.nn.functional as F
import numpy as np
import math
import os
import pandas as pd
import matplotlib.pyplot as plt


# Set seed.
seed = 42
np.random.seed(seed)
torch.manual_seed(seed)
torch.cuda.manual_seed(seed)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = True


SRC_LANGUAGE = 'en'
TGT_LANGUAGE = 'fr'

token_transform = {}
vocab_transform = {}
token_transform[SRC_LANGUAGE] = get_tokenizer('spacy', language='en_core_web_sm')
token_transform[TGT_LANGUAGE] = get_tokenizer('spacy', language='fr_core_news_sm')

csv = pd.read_csv("/content/eng_-french.csv",
    usecols=['English words/sentences', 'French words/sentences']
)
csv.head()
```

	English words/sentences	French words/sentences
0	Hi.	Salut!
1	Run!	Cours !
2	Run!	Courez !
3	Who?	Qui ?
4	Wow!	Ça alors !

New Section

New Section

```
train_csv, test_csv = train_test_split(csv, test_size=0.1)
```

```
# Custom Dataset class.
class TranslationDataset(Dataset):
    def __init__(self, csv):
        self.csv = csv

    def __len__(self):
        return len(self.csv)

    def __getitem__(self, idx):
        return(
            self.csv['English words/sentences'].iloc[idx],
            self.csv['French words/sentences'].iloc[idx]
        )


train_dataset = TranslationDataset(train_csv)
valid_dataset = TranslationDataset(test_csv)
iterator = iter(train_dataset)
print(next(iterator))
```

```
 ('They kept him waiting outside for a long time.', 'Ils le firent poireauter dehors.')
```

```
# Helper function to yield list of tokens.
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]])
# 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 vocab.
special_symbols = ['<unk>', '<pad>', '<bos>', '<eos>']
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
    # Create torchtext's Vocab object.
    vocab_transform[ln] = build_vocab_from_iterator(
        yield_tokens(train_dataset, ln),
        min_freq=1,
        specials=special_symbols,
        special_first=True,
    )
# Set ``UNK_IDX`` as the default index. This index is returned when the token is not found.
# If not set, it throws ``RuntimeError`` when the queried token is not found in the Vocabulary.
for ln in [SRC_LANGUAGE, TGT_LANGUAGE]:
    vocab_transform[ln].set_default_index(UNK_IDX)



# 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], # Numericalization
                                               tensor_transform) # Add BOS/EOS and create tensor
# function to collate data samples into batch tensors
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, batch_first=True)
    tgt_batch = pad_sequence(tgt_batch, padding_value=PAD_IDX, batch_first=True)
    return src_batch, tgt_batch
```

```
SRC_VOCAB_SIZE = len(vocab_transform[SRC_LANGUAGE])
TGT_VOCAB_SIZE = len(vocab_transform[TGT_LANGUAGE])
EMB_SIZE = 192
NHEAD = 6
FFN_HID_DIM = 192
BATCH_SIZE = 192
NUM_ENCODER_LAYERS = 3
NUM_DECODER_LAYERS = 3
DEVICE = 'cuda'
NUM_EPOCHS = 50


def generate_square_subsequent_mask(sz):
    mask = (torch.triu(torch.ones((sz, sz), device=DEVICE)) == 1).transpose(0, 1)
    mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
    return mask
def create_mask(src, tgt):
    src_seq_len = src.shape[1]
    tgt_seq_len = tgt.shape[1]
    tgt_mask = generate_square_subsequent_mask(tgt_seq_len)
    src_mask = torch.zeros((src_seq_len, src_seq_len),device=DEVICE).type(torch.bool)
    src_padding_mask = (src == PAD_IDX)
    tgt_padding_mask = (tgt == PAD_IDX)
    return src_mask, tgt_mask, src_padding_mask, tgt_padding_mask


class PositionalEncoding(nn.Module):
    def __init__(self, d_model, dropout, max_len=5000):
        """
        :param max_len: Input length sequence.
        :param d_model: Embedding dimension.
        :param dropout: Dropout value (default=0.1)
        """
        super(PositionalEncoding, self).__init__()
        self.dropout = nn.Dropout(p=dropout)
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2).float() * (-math.log(10000.0) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        pe = pe.unsqueeze(0)
        self.register_buffer('pe', pe)
    def forward(self, x):
        """
        Inputs of forward function
        :param x: the sequence fed to the positional encoder model (required).
        Shape:
            x: [sequence length, batch size, embed dim]
            output: [sequence length, batch size, embed dim]
        """
        x = x + self.pe[:, :x.size(1)]
        return self.dropout(x)


class TokenEmbedding(nn.Module):
    def __init__(self, vocab_size: int, emb_size):
        super(TokenEmbedding, self).__init__()
        self.embedding = nn.Embedding(vocab_size, emb_size)
        self.emb_size = emb_size
    def forward(self, tokens: Tensor):
        return self.embedding(tokens.long()) * math.sqrt(self.emb_size)
```

```
class Seq2SeqTransformer(nn.Module):
    def __init__(
        self,
        num_encoder_layers: int,
        num_decoder_layers: int,
        emb_size: int,
        nhead: int,
        src_vocab_size: int,
        tgt_vocab_size: int,
        dim_feedforward: int = 512,
        dropout: float = 0.1
    ):
        super(Seq2SeqTransformer, self).__init__()
        self.transformer = Transformer(
            d_model=emb_size,
            nhead=nhead,
            num_encoder_layers=num_encoder_layers,
            num_decoder_layers=num_decoder_layers,
            dim_feedforward=dim_feedforward,
            dropout=dropout,
            batch_first=True
        )
        self.generator = nn.Linear(emb_size, tgt_vocab_size)
        self.src_tok_emb = TokenEmbedding(src_vocab_size, emb_size)
        self.tgt_tok_emb = TokenEmbedding(tgt_vocab_size, emb_size)
        self.positional_encoding = PositionalEncoding(
            emb_size, dropout=dropout)
    def forward(self,
                src: Tensor,
                trg: Tensor,
                src_mask: Tensor,
                tgt_mask: Tensor,
                src_padding_mask: Tensor,
                tgt_padding_mask: Tensor,
                memory_key_padding_mask: Tensor):
        src_emb = self.positional_encoding(self.src_tok_emb(src))
        tgt_emb = self.positional_encoding(self.tgt_tok_emb(trg))
        outs = self.transformer(src_emb, tgt_emb, src_mask, tgt_mask, None,
                                src_padding_mask, tgt_padding_mask, memory_key_padding_mask)
        return self.generator(outs)
    def encode(self, src: Tensor, src_mask: Tensor):
        return self.transformer.encoder(self.positional_encoding(
                            self.src_tok_emb(src)), src_mask)
    def decode(self, tgt: Tensor, memory: Tensor, tgt_mask: Tensor):
        return self.transformer.decoder(self.positional_encoding(
                          self.tgt_tok_emb(tgt)), memory,
                          tgt_mask)


model = Seq2SeqTransformer(
    NUM_ENCODER_LAYERS,
    NUM_DECODER_LAYERS,
    EMB_SIZE,
    NHEAD,
    SRC_VOCAB_SIZE,
    TGT_VOCAB_SIZE,
    FFN_HID_DIM
).to(DEVICE)
# Total parameters and trainable parameters.
total_params = sum(p.numel() for p in model.parameters())
print(f"{total_params:,} total parameters.")
total_trainable_params = sum(
    p.numel() for p in model.parameters() if p.requires_grad)
print(f"{total_trainable_params:,} training parameters.")
print(model)
```

```
 14,487,719 total parameters.
 14,487,719 training parameters.
 Seq2SeqTransformer(
   (transformer): Transformer(
     (encoder): TransformerEncoder(
       (layers): ModuleList(
         (0-2): 3 x TransformerEncoderLayer(
           (self_attn): MultiheadAttention(
             (out_proj): NonDynamicallyQuantizableLinear(in_features=192, out_features=192, bias=True)
           )
           (linear1): Linear(in_features=192, out_features=192, bias=True)
           (dropout): Dropout(p=0.1, inplace=False)
```

```
            (linear2): Linear(in_features=192, out_features=192, bias=True)
            (norm1): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
            (norm2): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
            (dropout1): Dropout(p=0.1, inplace=False)
            (dropout2): Dropout(p=0.1, inplace=False)
          )
        )
        (norm): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
      )
      (decoder): TransformerDecoder(
        (layers): ModuleList(
          (0-2): 3 x TransformerDecoderLayer(
            (self_attn): MultiheadAttention(
              (out_proj): NonDynamicallyQuantizableLinear(in_features=192, out_features=192, bias=True)
            )
            (multihead_attn): MultiheadAttention(
              (out_proj): NonDynamicallyQuantizableLinear(in_features=192, out_features=192, bias=True)
            )
            (linear1): Linear(in_features=192, out_features=192, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
            (linear2): Linear(in_features=192, out_features=192, bias=True)
            (norm1): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
            (norm2): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
            (norm3): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
            (dropout1): Dropout(p=0.1, inplace=False)
            (dropout2): Dropout(p=0.1, inplace=False)
            (dropout3): Dropout(p=0.1, inplace=False)
          )
        )
        (norm): LayerNorm((192,), eps=1e-05, elementwise_affine=True)
      )
    )
    (generator): Linear(in_features=192, out_features=25319, bias=True)
    (src_tok_emb): TokenEmbedding(
      (embedding): Embedding(15389, 192)
    )
    (tgt_tok_emb): TokenEmbedding(
      (embedding): Embedding(25319, 192)
    )
    (positional_encoding): PositionalEncoding(
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )


loss_fn = torch.nn.CrossEntropyLoss(ignore_index=PAD_IDX)
optimizer = torch.optim.Adam(model.parameters(), lr=0.0001, betas=(0.9, 0.98), eps=1e-9)
```

```
train_dataloader = DataLoader(train_dataset, batch_size=BATCH_SIZE, collate_fn=collate_fn)
def train_epoch(model, optimizer):
    print('Training')
    model.train()
    losses = 0
    for src, tgt in tqdm(train_dataloader, total=len(list(train_dataloader))):
        # print(" ".join(vocab_transform[SRC_LANGUAGE].lookup_tokens(list(src[0].cpu().numpy()))).replace("<bos>", "").replace("
        # print(" ".join(vocab_transform[TGT_LANGUAGE].lookup_tokens(list(tgt[0].cpu().numpy()))).replace("<bos>", "").replace("
        src = src.to(DEVICE)
        tgt = tgt.to(DEVICE)

        tgt_input = tgt[:, :-1]

        src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)
        logits = model(
            src,
            tgt_input,
            src_mask,
            tgt_mask,
            src_padding_mask,
            tgt_padding_mask,
            src_padding_mask
        )
        optimizer.zero_grad()
        tgt_out = tgt[:, 1:]
        loss = loss_fn(logits.view(-1, TGT_VOCAB_SIZE), tgt_out.contiguous().view(-1))
        loss.backward()
        optimizer.step()
        losses += loss.item()
    return losses / len(list(train_dataloader))

val_dataloader = DataLoader(valid_dataset, batch_size=BATCH_SIZE, collate_fn=collate_fn)
def evaluate(model):
    print('Validating')
    model.eval()
    losses = 0
    for src, tgt in tqdm(val_dataloader, total=len(list(val_dataloader))):
        # print(" ".join(vocab_transform[SRC_LANGUAGE].lookup_tokens(list(src[0].cpu().numpy()))).replace("<bos>", "").replace("
        # print(" ".join(vocab_transform[TGT_LANGUAGE].lookup_tokens(list(tgt[0].cpu().numpy()))).replace("<bos>", "").replace("
        src = src.to(DEVICE)
        tgt = tgt.to(DEVICE)

        tgt_input = tgt[:, :-1]

        src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)

        logits = model(
            src,
            tgt_input,
            src_mask,
            tgt_mask,
            src_padding_mask,
            tgt_padding_mask,
            src_padding_mask
        )
        tgt_out = tgt[:, 1:]
        loss = loss_fn(logits.view(-1, TGT_VOCAB_SIZE), tgt_out.contiguous().view(-1))
        losses += loss.item()
    return losses / len(list(val_dataloader))



train_loss_list, valid_loss_list = [], []
for epoch in range(1, NUM_EPOCHS+1):
    start_time = timer()
    train_loss = train_epoch(model, optimizer)
    valid_loss = evaluate(model)
    end_time = timer()
    train_loss_list.append(train_loss)
    valid_loss_list.append(valid_loss)
    print((f"Epoch: {epoch}, Train loss: {train_loss:.3f}, Val loss: {valid_loss:.3f}, "f"Epoch time = {(end_time - start_time):
```

```
Training
100%                                          824/824 [01:55<00:00, 7.53it/s]
/usr/local/lib/python3.10/dist-packages/torch/nn/functional.py:5076: UserWarning: Support for mismatched key_padding_mask an
  warnings.warn(
Validating
100%                                          92/92 [00:06<00:00, 17.94it/s]
Epoch: 1, Train loss: 5.368, Val loss: 4.182, Epoch time = 173.613s

Training
100%                                          824/824 [01:54<00:00, 8.14it/s]
Validating
100%                                          92/92 [00:05<00:00, 17.85it/s]
Epoch: 2, Train loss: 3.945, Val loss: 3.537, Epoch time = 171.671s

Training
100%                                          824/824 [01:54<00:00, 8.17it/s]
Validating
100%                                          92/92 [00:06<00:00, 15.38it/s]
Epoch: 3, Train loss: 3.479, Val loss: 3.179, Epoch time = 172.184s

Training
100%                                          824/824 [01:55<00:00, 8.01it/s]
Validating
100%                                          92/92 [00:05<00:00, 17.99it/s]
Epoch: 4, Train loss: 3.177, Val loss: 2.924, Epoch time = 171.597s

Training
100%                                          824/824 [01:54<00:00, 8.04it/s]
Validating
100%                                          92/92 [00:06<00:00, 14.54it/s]
Epoch: 5, Train loss: 2.955, Val loss: 2.731, Epoch time = 173.028s

Training
100%                                          824/824 [01:54<00:00, 8.11it/s]
Validating
100%                                          92/92 [00:05<00:00, 17.94it/s]
Epoch: 6, Train loss: 2.773, Val loss: 2.568, Epoch time = 170.491s

Training
100%                                          824/824 [01:54<00:00, 8.02it/s]
Validating
100%                                          92/92 [00:06<00:00, 14.27it/s]
Epoch: 7, Train loss: 2.621, Val loss: 2.433, Epoch time = 171.539s

Training
100%                                          824/824 [01:54<00:00, 7.93it/s]
Validating
100%                                          92/92 [00:05<00:00, 18.29it/s]
Epoch: 8, Train loss: 2.491, Val loss: 2.316, Epoch time = 169.392s

Training
100%                                          824/824 [01:54<00:00, 8.03it/s]
Validating
100%                                          92/92 [00:05<00:00, 15.46it/s]
Epoch: 9, Train loss: 2.379, Val loss: 2.215, Epoch time = 171.518s

Training
100%                                          824/824 [01:54<00:00, 7.16it/s]
Validating
100%                                          92/92 [00:06<00:00, 18.06it/s]
Epoch: 10, Train loss: 2.281, Val loss: 2.131, Epoch time = 169.518s

Training
100%                                          824/824 [01:55<00:00, 7.96it/s]
Validating
100%                                          92/92 [00:05<00:00, 17.97it/s]
```

```
Epoch: 11, Train loss: 2.194, Val loss: 2.055, Epoch time = 172.981s

Training
100%                                        824/824 [01:55<00:00, 7.35it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.82it/s]
Epoch: 12, Train loss: 2.117, Val loss: 1.988, Epoch time = 171.542s

Training
100%                                        824/824 [01:55<00:00, 7.99it/s]
Validating
100%                                        92/92 [00:05<00:00, 14.06it/s]
Epoch: 13, Train loss: 2.048, Val loss: 1.933, Epoch time = 173.435s

Training
100%                                        824/824 [01:55<00:00, 7.75it/s]
Validating
100%                                        92/92 [00:05<00:00, 18.51it/s]
Epoch: 14, Train loss: 1.985, Val loss: 1.884, Epoch time = 170.818s

Training
100%                                        824/824 [01:54<00:00, 8.13it/s]
Validating
100%                                        92/92 [00:05<00:00, 15.03it/s]
Epoch: 15, Train loss: 1.928, Val loss: 1.837, Epoch time = 171.786s

Training
100%                                        824/824 [01:54<00:00, 7.06it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.45it/s]
Epoch: 16, Train loss: 1.877, Val loss: 1.794, Epoch time = 169.928s

Training
100%                                        824/824 [01:54<00:00, 8.06it/s]
Validating
100%                                        92/92 [00:05<00:00, 18.45it/s]
Epoch: 17, Train loss: 1.828, Val loss: 1.759, Epoch time = 170.736s

Training
100%                                        824/824 [01:54<00:00, 7.19it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.62it/s]
Epoch: 18, Train loss: 1.786, Val loss: 1.727, Epoch time = 170.130s

Training
100%                                        824/824 [01:54<00:00, 7.94it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.90it/s]
Epoch: 19, Train loss: 1.743, Val loss: 1.698, Epoch time = 170.031s

Training
100%                                        824/824 [01:54<00:00, 8.00it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.25it/s]
Epoch: 20, Train loss: 1.708, Val loss: 1.670, Epoch time = 171.693s

Training
100%                                        824/824 [01:55<00:00, 7.94it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.57it/s]
Epoch: 21, Train loss: 1.673, Val loss: 1.645, Epoch time = 172.275s

Training
100%                                        824/824 [01:55<00:00, 6.89it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.17it/s]
```

```
Epoch: 22, Train loss: 1.642, Val loss: 1.622, Epoch time = 172.466s

Training
100%                                        824/824 [01:55<00:00, 7.96it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.24it/s]
Epoch: 23, Train loss: 1.612, Val loss: 1.600, Epoch time = 173.272s

Training
100%                                        824/824 [01:55<00:00, 7.16it/s]
Validating
100%                                        92/92 [00:06<00:00, 18.12it/s]
Epoch: 24, Train loss: 1.585, Val loss: 1.583, Epoch time = 171.549s

Training
100%                                        824/824 [01:55<00:00, 7.88it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.77it/s]
Epoch: 25, Train loss: 1.558, Val loss: 1.566, Epoch time = 172.742s

Training
100%                                        824/824 [01:55<00:00, 7.00it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.86it/s]
Epoch: 26, Train loss: 1.533, Val loss: 1.551, Epoch time = 172.025s

Training
100%                                        824/824 [01:56<00:00, 8.01it/s]
Validating
100%                                        92/92 [00:05<00:00, 14.55it/s]
Epoch: 27, Train loss: 1.510, Val loss: 1.535, Epoch time = 174.407s

Training
100%                                        824/824 [01:56<00:00, 7.48it/s]
Validating
100%                                        92/92 [00:06<00:00, 17.81it/s]
Epoch: 28, Train loss: 1.488, Val loss: 1.524, Epoch time = 171.211s

Training
100%                                        824/824 [01:55<00:00, 8.03it/s]
Validating
100%                                        92/92 [00:05<00:00, 14.16it/s]
Epoch: 29, Train loss: 1.466, Val loss: 1.507, Epoch time = 173.390s

Training
100%                                        824/824 [01:55<00:00, 7.44it/s]
Validating
100%                                        92/92 [00:06<00:00, 18.25it/s]
Epoch: 30, Train loss: 1.447, Val loss: 1.495, Epoch time = 170.346s

Training
100%                                        824/824 [01:55<00:00, 7.90it/s]
Validating
100%                                        92/92 [00:05<00:00, 14.83it/s]
Epoch: 31, Train loss: 1.429, Val loss: 1.485, Epoch time = 172.923s

Training
100%                                        824/824 [01:55<00:00, 7.30it/s]
Validating
100%                                        92/92 [00:05<00:00, 17.56it/s]
Epoch: 32, Train loss: 1.411, Val loss: 1.473, Epoch time = 170.775s

Training
100%                                        824/824 [01:54<00:00, 8.03it/s]
Validating
100%                                        92/92 [00:05<00:00, 14.41it/s]
Epoch: 33, Train loss: 1.394, Val loss: 1.459, Epoch time = 172.513s
```

Epoch: 33, Train loss: 1.394, Val loss: 1.459, Epoch time = 172.513s

Training
100% 824/824 [01:55<00:00, 7.32it/s]
Validating
100% 92/92 [00:06<00:00, 17.79it/s]
Epoch: 34, Train loss: 1.378, Val loss: 1.453, Epoch time = 170.756s

Training
100% 824/824 [01:55<00:00, 7.89it/s]
Validating
100% 92/92 [00:05<00:00, 14.86it/s]
Epoch: 35, Train loss: 1.363, Val loss: 1.443, Epoch time = 173.327s

Training
100% 824/824 [01:55<00:00, 7.23it/s]
Validating
100% 92/92 [00:06<00:00, 17.76it/s]
Epoch: 36, Train loss: 1.349, Val loss: 1.436, Epoch time = 170.158s

Training
100% 824/824 [01:55<00:00, 8.01it/s]
Validating
100% 92/92 [00:05<00:00, 17.23it/s]
Epoch: 37, Train loss: 1.335, Val loss: 1.425, Epoch time = 172.025s

Training
100% 824/824 [01:55<00:00, 7.10it/s]
Validating
100% 92/92 [00:06<00:00, 18.00it/s]
Epoch: 38, Train loss: 1.321, Val loss: 1.415, Epoch time = 170.572s

Training
100% 824/824 [01:55<00:00, 7.94it/s]
Validating
100% 92/92 [00:05<00:00, 17.69it/s]
Epoch: 39, Train loss: 1.307, Val loss: 1.407, Epoch time = 171.765s

Training
100% 824/824 [01:56<00:00, 7.20it/s]
Validating
100% 92/92 [00:06<00:00, 17.61it/s]
Epoch: 40, Train loss: 1.295, Val loss: 1.402, Epoch time = 171.637s

Training
100% 824/824 [01:55<00:00, 7.91it/s]
Validating
100% 92/92 [00:05<00:00, 17.94it/s]
Epoch: 41, Train loss: 1.283, Val loss: 1.393, Epoch time = 172.271s

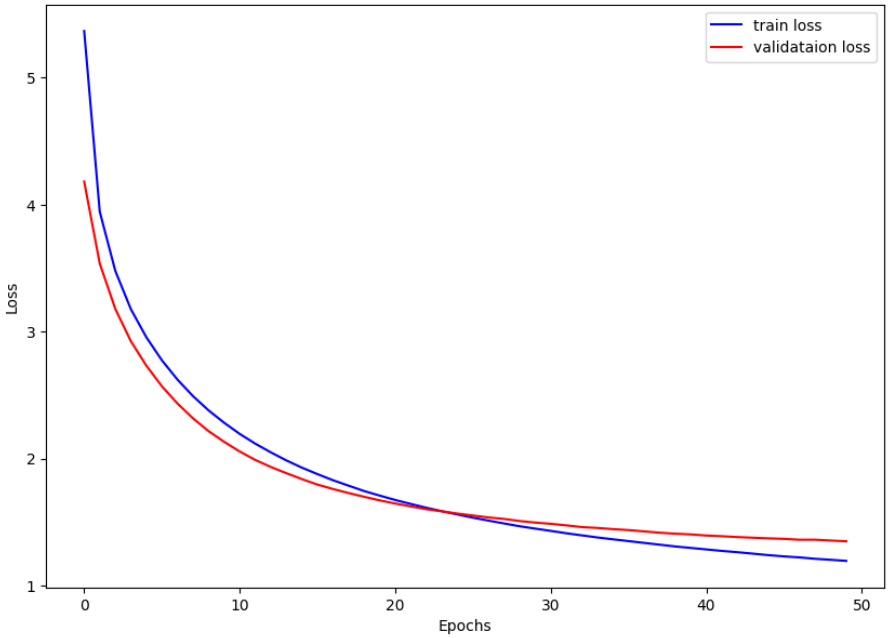
Training
100% 824/824 [01:55<00:00, 7.18it/s]
Validating
100% 92/92 [00:06<00:00, 17.14it/s]
Epoch: 42, Train loss: 1.272, Val loss: 1.387, Epoch time = 171.834s

Training
100% 824/824 [01:55<00:00, 7.91it/s]
Validating
100% 92/92 [00:05<00:00, 17.30it/s]
Epoch: 43, Train loss: 1.261, Val loss: 1.381, Epoch time = 173.072s

Training
100% 824/824 [01:57<00:00, 7.73it/s]
Validating
100% 92/92 [00:05<00:00, 17.62it/s]
Epoch: 44, Train loss: 1.250, Val loss: 1.376, Epoch time = 173.244s

```
Epoch: 44, Train loss: 1.230, Val loss: 1.370, Epoch time = 173.244s

Training
100%                                          824/824 [01:56<00:00, 8.00it/s]
Validating
100%                                          92/92 [00:06<00:00, 14.71it/s]
Epoch: 45, Train loss: 1.238, Val loss: 1.371, Epoch time = 173.829s

Training
100%                                          824/824 [01:56<00:00, 7.87it/s]
Validating
100%                                          92/92 [00:05<00:00, 16.53it/s]
Epoch: 46, Train loss: 1.228, Val loss: 1.367, Epoch time = 171.883s

Training
100%                                          824/824 [01:53<00:00, 8.11it/s]
Validating
100%                                          92/92 [00:05<00:00, 14.93it/s]
Epoch: 47, Train loss: 1.220, Val loss: 1.360, Epoch time = 171.486s

Training
100%                                          824/824 [01:54<00:00, 7.41it/s]
Validating
100%                                          92/92 [00:06<00:00, 17.98it/s]
Epoch: 48, Train loss: 1.209, Val loss: 1.360, Epoch time = 168.743s

Training
100%                                          824/824 [01:53<00:00, 8.14it/s]
Validating
100%                                          92/92 [00:05<00:00, 17.21it/s]
Epoch: 49, Train loss: 1.202, Val loss: 1.353, Epoch time = 169.973s

Training
100%                                          824/824 [01:53<00:00, 8.23it/s]
Validating
100%                                          92/92 [00:06<00:00, 15.56it/s]
Epoch: 50, Train loss: 1.193, Val loss: 1.348, Epoch time = 171.150s
```

```
os.makedirs('outputs', exist_ok=True)


def save_plots(train_loss, valid_loss):
    """
    Function to save the loss plots to disk.
    """
    # Loss plots.
    plt.figure(figsize=(10, 7))
    plt.plot(
        train_loss, color='blue', linestyle='-',
        label='train loss'
    )
    plt.plot(
        valid_loss, color='red', linestyle='-',
        label='validataion loss'
    )
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.savefig(os.path.join('outputs', 'loss.png'))
    plt.show()


save_plots(train_loss_list, valid_loss_list)
```




```
torch.save(model, 'outputs/model.pth')


model = torch.load('outputs/model.pth')


# Helper function to generate output sequence using greedy algorithm.
def greedy_decode(model, src, src_mask, max_len, start_symbol):
    src = src.to(DEVICE)
    src_mask = src_mask.to(DEVICE)
    memory = model.encode(src, src_mask)
    ys = torch.ones(1, 1).fill_(start_symbol).type(torch.long).to(DEVICE)
    for i in range(max_len-1):
        memory = memory.to(DEVICE)
        if i == 0:
            ys = ys.transpose(1, 0)
        tgt_mask = (generate_square_subsequent_mask(ys.size(1))
                    .type(torch.bool)).to(DEVICE)
        out = model.decode(ys, memory, tgt_mask)
        out = out
        prob = model.generator(out[:, -1])
        _, next_word = torch.max(prob, dim=1)
        next_word = next_word.item()
```