D:\myprojects\nlp\transformer\transformer_decoder.py

```python
# Decoder Only Transformer
#       (https://debuggercafe.com/text-generation-with-transformers/)
# conda install matplotlib
# conda install pytorch torchvision torchaudio pytorch-cuda=12.1 -c pytorch -c nvidia
#       또는 conda install pytorch torchvision torchaudio cpuonly -c pytorch
# set  KMP_DUPLICATE_LIB_OK=TRUE
# python transfor_decoder.py train 2

import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader
from collections import Counter

from tqdm import tqdm
import sys, math, warnings
import matplotlib.pyplot as plt

# Ignore Flash Attention compilation warnings
warnings.filterwarnings("ignore", message=".*not compiled with flash attention.*")
#=============================================================================
# [1/4] Data Preparation
#=============================================================================
SEQUENCE_LENGTH = 16    # 문장내 최대 단어 갯수 크기를 16으로 설정

with open('book.txt', 'r') as file:
    text = file.read()
# Tokenize the text into words
words = text.split()
word_counts = Counter(words)
vocab = list(word_counts.keys())
vocab_size = len(vocab)

word_to_int = {word: i for i, word in enumerate(vocab)}
int_to_word = {i: word for word, i in word_to_int.items()}
samples = [words[i:i+SEQUENCE_LENGTH+1] for i in range(len(words)-SEQUENCE_LENGTH)]

print(f"\nvocab : 총 [{vocab_size}] 단어 중 앞 10개\n",vocab[:10])
print("\nword_to_int[0~10]")
for i, word in enumerate(word_to_int):
    print(f"[{word}],",end='\t')
    if i==9 : break

#=============================================================================
# My Dataset Class
class TextDataset(Dataset):
    def __init__(self, samples, word_to_int):
        self.samples = samples
        self.word_to_int = word_to_int
    def __len__(self):
        return len(self.samples)
    def __getitem__(self, idx):
        sample = self.samples[idx]
            # 첫문자에서 마지막 문자까지 input vector 만들기
        input_seq = torch.LongTensor([self.word_to_int[word] for word in sample[:-1]])
            # 둘째 문자부터 마지막 문자까지 target vector 만들기
```

```
        target_seq = torch.LongTensor([self.word_to_int[word] for word in sample[1:]])
        return input_seq, target_seq

#================================================================================
#    Generate a square mask for the sequence. The masked positions are filled with float('-inf').
#    Unmasked positions are filled with float(0.0).
def generate_square_subsequent_mask(sz,device):
        # sz크기의 정방형 upper triangular matrix 후 transpose (filled with 1 in the upper triangle and 0 elsewhere)
    mask = (torch.triu(torch.ones(sz, sz, device=device)) == 1).transpose(0, 1)
        # 0의 값 --> -inf 로,  1의 값 --> 0으로 바꿈
    mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
    return mask

#================================================================================
# Embedding 된 입력 벡터에 Positional Encoding 값을 더하여 반환
class PositionalEncoding(nn.Module):
    def __init__(self, max_len, d_model, dropout=0.1):
        """
        :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)]  # word_encoding(x) + positional_encoding(x)
                # dropout(x)한 결과값 return
        return self.dropout(x)

#================================================================================
class TextGenDecoder(nn.Module):
    def __init__(self, vocab_size, embed_dim, num_layers, num_heads):
        super(TextGenDecoder, self).__init__()
        self.pos_encoder = PositionalEncoding(max_len=SEQUENCE_LENGTH, d_model=embed_dim)
        self.emb = nn.Embedding(vocab_size, embed_dim)
        self.decoder_layer = nn.TransformerDecoderLayer(
            d_model=embed_dim,
            nhead=num_heads,
            batch_first=True
        )
        self.decoder = nn.TransformerDecoder(
            decoder_layer=self.decoder_layer,
```

```
            num_layers=num_layers,
        )
        self.linear = nn.Linear(embed_dim, vocab_size)
        self.dropout = nn.Dropout(0.2)

    # Positional encoding is required. Else the model does not learn.
    def forward(self, x):
        emb = self.emb(x)

        # Generate input sequence mask with shape (SEQUENCE_LENGTH, SEQUENCE_LENGTH)
        input_mask = generate_square_subsequent_mask(x.size(1), x.device)

        x = self.pos_encoder(emb)
        x = self.decoder(x, memory=x, tgt_mask=input_mask, memory_mask=input_mask)
        x = self.dropout(x)
        out = self.linear(x)
        return out

#================================================================================
# draw loss changes
def draw_loss_changes(allEpochLosses):
    plt.plot(allEpochLosses, marker='o', linestyle='-', color='b')
    plt.title('Training Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid(True)
    plt.show()

#================================================================================
# Training
def trainDecoder(model, totalEpochs, dataloader, criterion, modelPath, printInterval=10):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    debugFlag = False
    if debugFlag :  # 각 파라미터가 gpu/cpu 중 뭘 사용하는지 확인
        for name, param in model.named_parameters():
            print(f"{name} is on {param.device}")

    model.train()   # 학습 모드로 설정

    print('\n','='*80,'\n*** Training Starting......')
    allEpochLosses = []
    totalDataSize = len(dataloader)
    for epochNo in tqdm(range(totalEpochs)):
        running_loss = 0
        for dataNo, (input_seq, target_seq) in enumerate(dataloader):
            input_seq, target_seq = input_seq.to(device), target_seq.to(device)

            outputs = model(input_seq)

            if epochNo == 0 and dataNo == 0  :
                print('-'*80)
                print(f'\nEpoch #[{epochNo}]')
                print('input 크기',input_seq.shape)
                print('input[:3] 값:',input_seq[:3])
                print('\ntarget 크기',target_seq.shape)
                print('target[:3] 값: ',target_seq[:3])
                print('\noutputs 크기: ', outputs.shape,'\n')

            target_seq = target_seq.contiguous().view(-1)
            outputs = outputs.view(-1, vocab_size)
```

```
            if dataNo % printInterval == 0  :
                print(f'\n---------- epochNo: [{epochNo}], dataNo [{dataNo} of {totalDataSize}] ----------')
                print('***outputs[0]:', outputs[0].detach().cpu().numpy())

            loss = criterion(outputs, target_seq.view(-1))
            if dataNo % printInterval == 0  :
                print(f'\t*** [{dataNo}]번째 data에 대한 loss:', loss.detach().cpu().numpy())

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            running_loss += loss.detach().cpu().numpy()

        epoch_loss = running_loss / len(dataloader)
        allEpochLosses.append(epoch_loss)
        print('\n','='*80)
        print(f"======== Epoch {epochNo} loss: {epoch_loss:.3f} ========")
        print('='*80,'\n')

    draw_loss_changes(allEpochLosses)
    torch.save(model.state_dict(), modelPath)

    return allEpochLosses

#================================================================================
# 입력 문장을 정수 벡터로 변환
def get_integer_vector(text):
    words = text.split()
    input_seq = torch.LongTensor([word_to_int[word] for word in words[-SEQUENCE_LENGTH:]]).unsqueeze(0)
    return input_seq

#================================================================================
# 예측 문장 생성
def sample_next(predictions):
    """
    Greedy sampling.
    """
    # Greedy approach.
    probabilities = F.softmax(predictions[:, -1, :], dim=-1).cpu()
    next_token = torch.argmax(probabilities)
    return int(next_token.cpu())

#================================================================================
# Prediction
def predict(model, query_sentence, totalPreditedWordNo):
    model.eval()

    sample = query_sentence   # query sentence로 시작
    for i in range(totalPreditedWordNo):
        int_vector = get_integer_vector(sample)
            # smaple 문장의 길이가, 가능한 최대 길이 보다 길어지면 종료
        if len(int_vector) >= SEQUENCE_LENGTH - 1:
            break
        input_tensor = int_vector.to(device)

        with torch.no_grad():   # prediction하여 출력값 계산
            predictions = model(input_tensor)

```

```
        next_token_ndx = sample_next(predictions)   # 최대 출력값 인덱스 계산
        sample += ' ' + int_to_word[next_token_ndx] # 최대 출력 단어를 sample에 추가
    return(sample)

#=============================================================================
# Model feature printing
def displayModelFeature(model) :
    if showModelFlag :
        print('-'*80)
        print('Decoder Model 구조')
        print(model)
        print('-'*80)

        # Total parameters and trainable parameters.
        total_params = sum(p.numel() for p in model.parameters())
        print(f"\n{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.\n")

#=============================================================================
# Gpu status printing
def displayGpuStatus(showFlag):
    totalGpuNum = torch.cuda.device_count()

    if showFlag :
        print("="*80)
        if totalGpuNum  > 0 :
            if torch.cuda.is_available():
                torch.cuda.empty_cache()
            print("Available GPU Devices: ")
            for i in range(totalGpuNum):
                print("[[[ Device [%d]: %s ]]]" %(i, torch.cuda.get_device_properties(i)))
            print("-"*50)

            print("현재 사용중인 GPU #: ",torch.cuda.current_device())
            print('Allocated:', round(torch.cuda.memory_allocated(0)/1024**3,1), 'GB')
            print('Cached:   ', round(torch.cuda.memory_reserved(0)/1024**3,1), 'GB')
        else :
            print("NO Gpu exists..... Will use the CPU")
        print("="*80)

    return totalGpuNum

#=============================================================================
# [2/4] dataset 준비
#=============================================================================
BATCH_SIZE = 32
dataset = TextDataset(samples, word_to_int)
dataloader = DataLoader(
    dataset,
    batch_size=BATCH_SIZE,
    shuffle=True,
)
print('\n*** input, target dataset[0]:\n',dataset[0])
print('\n*** input, target dataset[1]:\n',dataset[1])

modelPath = './decoder.pth'

```

```python
#================================================================================
# [3/4] Decoder 모델 준비
#================================================================================
totalEpochs = 100
learning_rate = 0.001
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f'\n*** device = {device} \n')

printInterval = 10 if device == 'cpu' else 100

model = TextGenDecoder(
    vocab_size=vocab_size,
    embed_dim=512,
    num_layers=6,
    num_heads=4,
).to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

showModelFlag = True
if showModelFlag == True:
    displayModelFeature(model)


#================================================================================
# [4/4] main, train or predict
#================================================================================
if __name__ == "__main__" :
    print('\n','='*80)
    print('Usage [Training]: python decoder.py   train   totalEpochNo[예:10]')
    print('Usage [Prediction]: python decoder.py   predict   totalPreditedWordNo[예:100]')
    print('*** ',displayGpuStatus(True),' Gpu(s) will be used.\n')

    if len(sys.argv) >= 3 :
            #-------------------------------- 학습
        if 'train' in sys.argv[1] and int(sys.argv[2])> 0:
            totalEpochs = int(sys.argv[2])
            allEpochLosses = trainDecoder(model, totalEpochs, dataloader, criterion,
modelPath, printInterval)

            # Test prediction using a sentence
        elif 'predict' in sys.argv[1] and int(sys.argv[2])> 0:
            totalPreditedWordNo = int(sys.argv[2])
            state_dict = torch.load(modelPath)
            # Update the current model state with the loaded state dictionary
            model.load_state_dict(state_dict)
            model = model.to(device)
            print("*** Model loaded successfully from ", modelPath, '\n')

            if showModelFlag :
                displayModelFeature(model)

            while True:
                query_sentence = input("\n*** Type a sentence or 'quit' to stop: ")
                if 'quit' in query_sentence :
                    break
                #print("Predicting for sentence:", test_sentence)
                print("===> query sentence:", query_sentence)
                predicted_output = predict(model, query_sentence, totalPreditedWordNo)
```

```
352                 print("===> Predicted output:", predicted_output, '\n')
353
```