Artificial Intelligence Lab Work (6)

レポート解答用紙（Report Answer Sheet）

学生証番号（Student ID): 20521150

名前(Name): Pham Quoc Cuong

問題 1.

(プログラム)

```
import requests
import torch
import torch.nn.functional as F
import torchtext
import tarfile

url = "https://nlp.stanford.edu/projects/nmt/data/iwslt15.en-vi/"
train_en = [line.split() for line in requests.get(url+"train.en").text.splitlines()]
train_vi = [line.split() for line in requests.get(url+"train.vi").text.splitlines()]
test_en = [line.split() for line in requests.get(url+"tst2013.en").text.splitlines()]
test_vi = [line.split() for line in requests.get(url+"tst2013.vi").text.splitlines()]

MODELNAME = "iwslt15-en-vi-rnn.model"
EPOCH = 10
BATCHSIZE = 128
LR = 0.0001
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"

def make_vocab(train_data, min_freq):
  vocab = {}
  for tokenlist in train_data:
    for token in tokenlist:
      if token not in vocab:
        vocab[token] = 0
      vocab[token] += 1
  vocablist = [('<unk>', 0), ('<pad>', 0), ('<cls>', 0), ('<eos>', 3)]
  vocabidx = {}
  for token, freq in vocab.items():
    if freq >= min_freq:
      idx = len(vocablist)
```

```
      vocablist.append((token, freq))
      vocabidx[token] = idx
  vocabidx['<unk>'] = 0
  vocabidx['<pad>'] = 1
  vocabidx['<cls>'] = 2
  vocabidx['<eos>'] = 3
  return vocablist, vocabidx

vocablist_en, vocabidx_en = make_vocab(train_en, 3)
vocablist_vi, vocabidx_vi = make_vocab(train_vi, 3)

def preprocess(data, vocabidx):
  rr = []
  for tokenlist in data:
    tkl = ['<cls>']
    for token in tokenlist:
      tkl.append(token if token in vocabidx else '<unk>')
    tkl.append('<eos>')
    rr.append((tkl))
  return rr

train_en_prep = preprocess(train_en, vocabidx_en)
train_vi_prep = preprocess(train_vi, vocabidx_vi)
test_en_prep = preprocess(test_en, vocabidx_en)

train_data = list(zip(train_en_prep, train_vi_prep))
train_data.sort(key = lambda x: (len(x[0]), len(x[1])))
test_data = list(zip(test_en_prep, test_en, test_vi))

def make_batch(data, batchsize):
  bb = []
  ben = []
  bvi = []
  for en, vi in data:
    ben.append(en)
    bvi.append(vi)
    if len(ben) >= batchsize:
      bb.append((ben, bvi))
```

```
      ben = []
      bvi = []
  if len(ben) > 0:
    bb.append((ben, bvi))
  return bb

train_data = make_batch(train_data, BATCHSIZE)

def padding_batch(b):
  maxlen = max([len(x) for x in b])
  for tkl in b:
    for i in range(maxlen - len(tkl)):
      tkl.append('<pad>')

def padding(bb):
  for ben, bvi in bb:
    padding_batch(ben)
    padding_batch(bvi)

padding(train_data)

train_data = [([[vocabidx_en[token] for token in tokenlist] for tokenlist in ben],
             [[vocabidx_vi[token] for token in tokenlist] for tokenlist in bvi]) for
ben, bvi in train_data]
test_data = [([vocabidx_en[token] for token in enprep], en, vi) for enprep, en, vi in
test_data]

class RNNEncDec(torch.nn.Module):
    def __init__(self, vocablist_x, vocabidx_x, vocablist_y, vocabidx_y):
        super(RNNEncDec, self).__init__()
        self.encemb   =   torch.nn.Embedding(len(vocablist_x),   300,   padding_idx   =
vocabidx_x['<pad>'])
        self.encrnn = torch.nn.Linear(300, 300)
        self.decemb   =   torch.nn.Embedding(len(vocablist_x),   300,   padding_idx   =
vocabidx_y['<pad>'])
        self.decrnn = torch.nn.Linear(300, 300)
        self.decout = torch.nn.Linear(300, len(vocablist_y))
```

```
    def forward(self,x):
        x, y = x[0], x[1]
        #enc
        e_x = self.encemb(x)
        n_x = e_x.size()[0]
        h = torch.zeros(300, dtype=torch.float32).to(DEVICE)
        for i in range(n_x):
          h = F.relu(e_x[i] + self.encrnn(h))
        #dec
        e_y = self.decemb(y)
        n_y = e_y.size()[0]
        loss = torch.tensor(0., dtype=torch.float32).to(DEVICE)
        for i in range(n_y-1):
            h = F.relu(e_y[i] + self.decrnn(h))
            loss += F.cross_entropy(self.decout(h), y[i+1])
        return loss

    def evaluate(self, x, vocablist_y, vocabidx_y):
        # encoder
        #推論は1文ずつ行うので、 x には文長xバッチサイズ 1 のミニバッチが入っている。
        e_x = self.encemb(x)
        n_x = e_x.size()[0]
        #エンコーダー部は forward とほぼ同じ。
        h = torch.zeros(300, dtype = torch.float32).to(DEVICE)
        for i in range(n_x):
            h = F.relu(e_x[i] + self.encrnn(h))
        # decoder
        #デコーダーの入力(バッチサイズ 1)を作る。最初はトークンを入力する
        y = torch.tensor([vocabidx_y['<cls>']]).to(DEVICE)
        e_y = self.decemb(y)
        pred = []
        for i in range(30):
            h = F.relu(e_y + self.decrnn(h))
            pred_id = self.decout(h).squeeze().argmax()
            #pred_id が予測する出力単語 ID pred_id がの ID と等しければ推論終了
            if pred_id == vocabidx_y['<eos>']:
                break
            pred_y = vocablist_y[pred_id][0]
```

```
            pred.append(pred_y)
            #デコーダーは 1 単語ずつ処理をし、得られた出力を次の入力とする
            y[0] = pred_id
            e_y = self.decemb(y)

        return pred

def train():
  model = RNNEncDec(vocablist_en, vocabidx_en, vocablist_vi, vocabidx_vi).to(DEVICE)
  optimizer = torch.optim.Adam(model.parameters(), lr = LR)
  for epoch in range(EPOCH):
    loss = 0
    step = 0
    for ben, bvi in train_data:
      ben = torch.tensor(ben, dtype=torch.int64).transpose(0,1).to(DEVICE)
      bvi = torch.tensor(bvi, dtype=torch.int64).transpose(0,1).to(DEVICE)
      optimizer.zero_grad()
      batchloss = model((ben, bvi))
      batchloss.backward()
      optimizer.step()
      loss = loss + batchloss.item()
      if step % 100 == 0:
        print("step:", step, "batchloss:", batchloss.item())
      step += 1

    print("epoch", epoch, ": loss", loss)
  torch.save(model.state_dict(), MODELNAME)

def test():
    total = 0
    correct = 0
    model = RNNEncDec(vocablist_en, vocabidx_en, vocablist_vi, vocabidx_vi).to(DEVICE)
    model.load_state_dict(torch.load(MODELNAME))
    model.eval()
    ref = []
    pred = []
    for enprep, en, vi in test_data:
        input = torch.tensor([enprep], dtype=torch.int64).transpose(0, 1).to(DEVICE)
```

```
        p = model.evaluate(input, vocablist_vi, vocabidx_vi)
        print("INPUT", en)
        print("REF", vi)
        print("MT", p)
        ref.append([vi])
        pred.append(p)
    bleu = torchtext.data.metrics.bleu_score(pred, ref)
    print("total:", len(test_data))
    print("bleu:", bleu)

train()
test()
```

（実行結果）

```
step: 0 batchloss: 27.172576904296875
step: 100 batchloss: 58.17341613769531
step: 200 batchloss: 74.98722076416016
step: 300 batchloss: 92.24523162841797
step: 400 batchloss: 122.39286041259766
step: 500 batchloss: 107.60729217529297
step: 600 batchloss: 125.73492431640625
step: 700 batchloss: 181.92393493652344
step: 800 batchloss: 191.50404357910156
step: 900 batchloss: 249.77056884765625
step: 1000 batchloss: 307.3482971191406
epoch 0 : loss 166257.83545303345
step: 0 batchloss: 14.811253547668457
step: 100 batchloss: 47.186012268066406
step: 200 batchloss: 63.199371337890625
step: 300 batchloss: 79.02168273925781
step: 400 batchloss: 106.95979309082031
step: 500 batchloss: 96.59516143798828
step: 600 batchloss: 112.09869384765625
step: 700 batchloss: 165.00559997558594
step: 800 batchloss: 175.23382568359375
step: 900 batchloss: 224.41159057617188
step: 1000 batchloss: 287.87225341796875
epoch 1 : loss 137898.50154399872
```

```
step: 0 batchloss: 13.23542308807373
step: 100 batchloss: 45.3863525390625
step: 200 batchloss: 60.69123077392578
step: 300 batchloss: 75.49849700927734
step: 400 batchloss: 102.2755126953125
step: 500 batchloss: 93.99681854248047
step: 600 batchloss: 111.10321807861328
step: 700 batchloss: 158.58163452148438
step: 800 batchloss: 171.3170166015625
step: 900 batchloss: 217.24542236328125
step: 1000 batchloss: 280.5041809082031
epoch 2 : loss 135461.8858499527
step: 0 batchloss: 8.008686065673828
step: 100 batchloss: 43.066532135009766
step: 200 batchloss: 57.8975830078125
step: 300 batchloss: 72.42780303955078
step: 400 batchloss: 98.28285217285156
step: 500 batchloss: 89.97473907470703
step: 600 batchloss: 104.9176254272461
step: 700 batchloss: 153.31304931640625
step: 800 batchloss: 164.0221405029297
step: 900 batchloss: 210.53587341308594
step: 1000 batchloss: 273.4266662597656
epoch 3 : loss 128628.151927948
```

```
step: 0 batchloss: 8.20518970489502
step: 100 batchloss: 41.52042007446289
step: 200 batchloss: 55.829383850097656
step: 300 batchloss: 70.00132751464844
step: 400 batchloss: 94.96910095214844
step: 500 batchloss: 87.28950500488281
step: 600 batchloss: 102.02731323242188
step: 700 batchloss: 149.25927734375
step: 800 batchloss: 159.74171447753906
step: 900 batchloss: 205.62513732910156
step: 1000 batchloss: 267.88970947265625
epoch 4 : loss 125253.09456348419
step: 0 batchloss: 8.290722846984863
step: 100 batchloss: 40.48969268798828
step: 200 batchloss: 54.33784866333008
step: 300 batchloss: 68.18540954589844
step: 400 batchloss: 92.57718658447266
step: 500 batchloss: 85.52182006835938
step: 600 batchloss: 99.98380279541016
step: 700 batchloss: 146.23585510253906
step: 800 batchloss: 156.51971435546875
step: 900 batchloss: 201.97161865234375
step: 1000 batchloss: 263.6092834472656
epoch 5 : loss 122754.63380908966
```

```
step: 0 batchloss: 7.959069728851318
step: 100 batchloss: 39.57741165161133
step: 200 batchloss: 53.24683380126953
step: 300 batchloss: 66.73372650146484
step: 400 batchloss: 90.54024505615234
step: 500 batchloss: 83.94981384277344
step: 600 batchloss: 98.80986785888672
step: 700 batchloss: 143.8058624267578
step: 800 batchloss: 161.8977813720703
step: 900 batchloss: 201.3871307373047
step: 1000 batchloss: 261.9737548828125
epoch 6 : loss 122993.73261213303
step: 0 batchloss: 6.338806629180908
step: 100 batchloss: 39.216468811035156
step: 200 batchloss: 52.95828628540039
step: 300 batchloss: 66.50444793701172
step: 400 batchloss: 89.99591064453125
step: 500 batchloss: 83.53227233886719
step: 600 batchloss: 97.41558074951172
step: 700 batchloss: 142.3235626220703
step: 800 batchloss: 152.84986877441406
step: 900 batchloss: 197.0547332763672
step: 1000 batchloss: 258.2327575683594
epoch 7 : loss 119612.07580900192
```

```
step: 0 batchloss: 6.573685646057129
step: 100 batchloss: 38.300811767578125
step: 200 batchloss: 51.768367767333984
step: 300 batchloss: 65.09220123291016
step: 400 batchloss: 87.99906921386719
step: 500 batchloss: 82.02564239501953
step: 600 batchloss: 96.29276275634766
step: 700 batchloss: 140.44273376464844
step: 800 batchloss: 150.37867736816406
step: 900 batchloss: 194.5476837158203
step: 1000 batchloss: 255.48544311523438
epoch 8 : loss 117837.4348745346
step: 0 batchloss: 6.687434673309326
step: 100 batchloss: 37.85627746582031
step: 200 batchloss: 51.157405853271484
step: 300 batchloss: 64.23204803466797
step: 400 batchloss: 86.61820220947266
step: 500 batchloss: 81.02442932128906
step: 600 batchloss: 95.31536102294922
step: 700 batchloss: 139.04476928710938
step: 800 batchloss: 148.6105499267578
step: 900 batchloss: 192.8077850341797
step: 1000 batchloss: 253.1693572998047
epoch 9 : loss 116563.62330007553
```

INPUT ['When', 'I', 'was', 'little', ',', 'I', 'thought', 'my', 'country', 'was', 'the', 'best', 'on', 'the', 'planet', ',', 'and', 'I', 'grew', 'up', 'singing', 'a', 'song', 'called', '"', 'Nothing', 'To', 'Envy', '.', '"']
REF ['Khi', 'tôi', 'còn', 'nhỏ', ',', 'Tôi', 'nghĩ', 'rằng', 'BắcTriều', 'Tiên', 'là', 'đất', 'nước', 'tốt', 'nhất', 'trên', 'thế', 'giới', 'và', 'tôi', 'thường', 'hát', 'bài', '"', 'Chúng', 'ta', 'chẳng', 'có', 'gì', 'phải', 'ghen', 'tị', '.', '"']
MT ['Và', 'tôi', 'nghĩ', 'rằng', 'tôi', 'đã', 'nói', '"', '"', '"', '"', '"', '"', '"', '"', '"', '"', '"', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>']
INPUT ['And', 'I', 'was', 'very', 'proud', '.']
REF ['Tôi', 'đã', 'rất', 'tự', 'hào', 'về', 'đất', 'nước', 'tôi', '.']
MT ['Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', '"', '"', '"', '"', '"', '"', '"', '"', '"', '"', '"', '"']
INPUT ['In', 'school', ',', 'we', 'spent', 'a', 'lot', 'of', 'time', 'studying', 'the', 'history', 'of', 'Kim', 'Il-Sung', ',', 'but', 'we', 'never', 'learned', 'much', 'about', 'the', 'outside', 'world', ',', 'except', 'that', 'America', ',', 'South', 'Korea', ',', 'Japan', 'are', 'the', 'enemies', '.']
REF ['Ở', 'trường', ',', 'chúng', 'tôi', 'dành', 'rất', 'nhiều', 'thời', 'gian', 'để', 'học', 'về', 'cuộc', 'đời', 'của', 'chủ', 'tịch', 'Kim', 'II-', 'Sung', ',', 'nhưng', 'lại', 'không', 'học', 'nhiều', 'về', 'thế', 'giới', 'bên', 'ngoài', ',', 'ngoại', 'trừ', 'việc', 'Hoa', 'Kỳ', ',', 'Hàn', 'Quốc', 'và', 'Nhật', 'Bản', 'là', 'kẻ', 'thù', 'của', 'chúng', 'tôi', '.']
MT ['Nó', 'là', 'một', 'người', 'đàn', 'ông', ',', 'tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', '"', '"', '"', '"', '"']

INPUT ['Thank', 'you', 'very', 'much', '.']
REF ['Cảm', 'ơn', 'rất', 'nhiều', '.']
MT ['Chúng', 'ta', 'có', 'thể', 'làm', 'được', 'điều', 'này', ',', 'tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', 'rằng', '"', 'Tôi', 'đã', 'nói', '"', '"', '"']
total: 1268
bleu: 0.003965999465435743

問題 2.

(プログラム)

```
MODELNAME = "iwslt15-en-vi-lstm.model"
EPOCH = 10
BATCHSIZE = 128
LR = 0.0001
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"

class LSTM(torch.nn.Module):
    def __init__(self, vocablist_x, vocabidx_x, vocablist_y, vocabidx_y):
        super(LSTM, self).__init__()

        self.encemb    =    torch.nn.Embedding(len(vocablist_x),    256,    padding_idx    =
vocabidx_x['<pad>'])
        self.dropout = torch.nn.Dropout(0.5)
        self.enclstm = torch.nn.LSTM(256,516,2,dropout=0.5)

        self.decemb    =    torch.nn.Embedding(len(vocablist_x),    256,    padding_idx    =
vocabidx_y['<pad>'])
        self.declstm = torch.nn.LSTM(256,516,2,dropout=0.5)
        self.decout = torch.nn.Linear(516, len(vocabidx_y))

    def forward(self,x):
        x, y = x[0], x[1]
        e_x = self.dropout(self.encemb(x))

        outenc,(hidden,cell) = self.enclstm(e_x)

        n_y=y.shape[0]
        outputs = torch.zeros(n_y,BATCHSIZE,len(vocablist_vi)).to(DEVICE)
        loss = torch.tensor(0.,dtype=torch.float32).to(DEVICE)
        for i in range(n_y-1):
            input = y[i]
            input = input.unsqueeze(0)
            input = self.dropout(self.decemb(input))
            outdec, (hidden,cell) = self.declstm(input,(hidden,cell))
            output = self.decout(outdec.squeeze(0))
            input = y[i+1]
```

```
            loss += F.cross_entropy(output, y[i+1])
        return loss

    def evaluate(self,x,vocablist_y,vocabidx_y):
        e_x = self.dropout(self.encemb(x))
        outenc,(hidden,cell)=self.enclstm(e_x)

        y = torch.tensor([vocabidx_y['<cls>']]).to(DEVICE)
        pred=[]
        for i in range(30):
            input = y
            input = input.unsqueeze(0)
            input = self.dropout(self.decemb(input))
            outdec,(hidden,cell)= self.declstm(input,(hidden,cell))
            output = self.decout(outdec.squeeze(0))
            pred_id = output.squeeze().argmax().item()
            if pred_id == vocabidx_y['<eos>']:
                break
            pred_y = vocablist_y[pred_id][0]
            pred.append(pred_y)
            y[0]=pred_id
            input=y
        return pred

def train():
    model = LSTM(vocablist_en, vocabidx_en, vocablist_vi, vocabidx_vi).to(DEVICE)
    optimizer = torch.optim.Adam(model.parameters(), lr = LR)
    for epoch in range(EPOCH):
        loss = 0
        step = 0
       for ben, bvi in train_data:
            ben = torch.tensor(ben, dtype=torch.int64).transpose(0,1).to(DEVICE)
            bvi = torch.tensor(bvi, dtype=torch.int64).transpose(0,1).to(DEVICE)
            optimizer.zero_grad()
            batchloss = model((ben, bvi))
            batchloss.backward()
            optimizer.step()
            loss = loss + batchloss.item()
```

```
            if step % 100 == 0:
                print("step:", step, "batchloss:", batchloss.item())
            step += 1

        print("epoch", epoch, ": loss", loss)
    torch.save(model.state_dict(), MODELNAME)

def test():
    total = 0
    correct = 0
    model = LSTM(vocablist_en, vocabidx_en, vocablist_vi, vocabidx_vi).to(DEVICE)
    model.load_state_dict(torch.load(MODELNAME))
    model.eval()
    ref = []
    pred = []
    for enprep, en, vi in test_data:
        input = torch.tensor([enprep], dtype=torch.int64).transpose(0, 1).to(DEVICE)
        p = model.evaluate(input, vocablist_vi, vocabidx_vi)
        print("INPUT", en)
        print("REF", vi)
        print("MT", p)
        ref.append([vi])
        pred.append(p)
    bleu = torchtext.data.metrics.bleu_score(pred, ref)
    print("total:", len(test_data))
    print("bleu:", bleu)

train()
test()
```

（実行結果）

```
step: 0 batchloss: 27.84752655029297
step: 100 batchloss: 58.818382263183594
step: 200 batchloss: 77.45357513427734
step: 300 batchloss: 95.80945587158203
step: 400 batchloss: 127.5394287109375
step: 500 batchloss: 112.57477569580078
step: 600 batchloss: 130.96348571777344
step: 700 batchloss: 200.33200073242188
step: 800 batchloss: 206.8363494873047
step: 900 batchloss: 267.7705078125
step: 1000 batchloss: 331.2012939453125
epoch 0 : loss 165335.60354614258
step: 0 batchloss: 8.255011558532715
step: 100 batchloss: 51.3633918762207
step: 200 batchloss: 70.18926239013672
step: 300 batchloss: 88.05742645263672
step: 400 batchloss: 118.88196563720703
step: 500 batchloss: 104.76687622070312
step: 600 batchloss: 122.68170166015625
step: 700 batchloss: 186.54640197753906
step: 800 batchloss: 194.05775451660156
step: 900 batchloss: 251.36830139160156
step: 1000 batchloss: 314.6599426269531
epoch 1 : loss 152213.5560464859
```

```
step: 0 batchloss: 5.39594841003418
step: 100 batchloss: 47.97709274291992
step: 200 batchloss: 65.75201416015625
step: 300 batchloss: 82.76996612548828
step: 400 batchloss: 112.3674545288086
step: 500 batchloss: 99.29512023925781
step: 600 batchloss: 115.75562286376953
step: 700 batchloss: 176.84986877441406
step: 800 batchloss: 184.86196899414062
step: 900 batchloss: 239.4592742919922
step: 1000 batchloss: 302.5055236816406
epoch 2 : loss 144549.3693523407
step: 0 batchloss: 4.059388160705566
step: 100 batchloss: 44.8297119140625
step: 200 batchloss: 61.785884857177734
step: 300 batchloss: 78.34351348876953
step: 400 batchloss: 106.91899871826172
step: 500 batchloss: 94.87004089355469
step: 600 batchloss: 110.57605743408203
step: 700 batchloss: 169.21096801757812
step: 800 batchloss: 177.14468383789062
step: 900 batchloss: 230.16114807128906
step: 1000 batchloss: 292.9637145996094
epoch 3 : loss 138461.77570819855
```

```
step: 0 batchloss: 2.9176273345947266
step: 100 batchloss: 42.613895416259766
step: 200 batchloss: 59.20978546142578
step: 300 batchloss: 74.96353149414062
step: 400 batchloss: 102.92110443115234
step: 500 batchloss: 91.0498275756836
step: 600 batchloss: 106.95613861083984
step: 700 batchloss: 163.57997131347656
step: 800 batchloss: 171.8434600830078
step: 900 batchloss: 222.83396911621094
step: 1000 batchloss: 285.4816589355469
epoch 4 : loss 133813.55984210968
step: 0 batchloss: 2.1207430362701416
step: 100 batchloss: 40.828102111816406
step: 200 batchloss: 56.99211883544922
step: 300 batchloss: 72.32482147216797
step: 400 batchloss: 99.77446746826172
step: 500 batchloss: 88.63948822021484
step: 600 batchloss: 104.14561462402344
step: 700 batchloss: 158.75535583496094
step: 800 batchloss: 167.23985290527344
step: 900 batchloss: 216.93402099609375
step: 1000 batchloss: 278.63897705078125
epoch 5 : loss 130159.01452565193
```

```
step: 0 batchloss: 1.8184490203857422
step: 100 batchloss: 39.957191467285156
step: 200 batchloss: 55.559303283691406
step: 300 batchloss: 70.17080688476562
step: 400 batchloss: 97.01287841796875
step: 500 batchloss: 86.0363540649414
step: 600 batchloss: 101.59703826904297
step: 700 batchloss: 154.16770935058594
step: 800 batchloss: 163.6635284423828
step: 900 batchloss: 211.86846923828125
step: 1000 batchloss: 273.8332214355469
epoch 6 : loss 127060.66787815094
step: 0 batchloss: 1.731295108795166
step: 100 batchloss: 38.821075439453125
step: 200 batchloss: 54.00948715209961
step: 300 batchloss: 68.56170654296875
step: 400 batchloss: 94.90394592285156
step: 500 batchloss: 84.66358184814453
step: 600 batchloss: 99.26224517822266
step: 700 batchloss: 151.16839599609375
step: 800 batchloss: 159.58309936523438
step: 900 batchloss: 207.49203491210938
step: 1000 batchloss: 269.2329406738281
epoch 7 : loss 124451.63530302048
```

```
step: 0 batchloss: 1.6039631366729736
step: 100 batchloss: 37.92952346801758
step: 200 batchloss: 52.65158462524414
step: 300 batchloss: 66.8499984741211
step: 400 batchloss: 92.98161315917969
step: 500 batchloss: 82.87738037109375
step: 600 batchloss: 97.40608978271484
step: 700 batchloss: 148.78904724121094
step: 800 batchloss: 156.80447387695312
step: 900 batchloss: 204.0107421875
step: 1000 batchloss: 266.12994384765625
epoch 8 : loss 122157.00905394554
step: 0 batchloss: 1.5359781980514526
step: 100 batchloss: 36.9404411315918
step: 200 batchloss: 51.44622802734375
step: 300 batchloss: 65.43421936035156
step: 400 batchloss: 90.65040588378906
step: 500 batchloss: 81.61491394042969
step: 600 batchloss: 96.09838104248047
step: 700 batchloss: 146.2509765625
step: 800 batchloss: 154.83839416503906
step: 900 batchloss: 201.165283203125
step: 1000 batchloss: 262.0869140625
epoch 9 : loss 120115.09836804867
```

Test function result:

```
INPUT ['When', 'I', 'was', 'little', ',', 'I', 'thought', 'my', 'country', 'was', 'the', 'best', 'on', 'the', 'planet', ',', 'an
d', 'I', 'grew', 'up', 'singing', 'a', 'song', 'called', '&quot;', 'Nothing', 'To', 'Envy', '.', '&quot;']
REF ['Khi', 'tôi', 'còn', 'nhỏ', ',', 'Tôi', 'nghĩ', 'rằng', 'BắcTriều', 'Tiên', 'là', 'đất', 'nước', 'tốt', 'nhất', 'trên', 'th
ế', 'giới', 'và', 'tôi', 'thường', 'hát', 'bài', '&quot;', 'Chúng', 'ta', 'chẳng', 'có', 'gì', 'phải', 'ghen', 'tị', '.', '&quo
t;']
MT ['Khi', 'tôi', 'đã', 'nói', ',', '&quot;', 'Tôi', 'đã', 'nói', 'về', 'một', 'vài', 'người', 'trong', 'những', 'người', 'phụ',
'nữ', 'và', 'nói', 'rằng', '&quot;', 'Tôi', 'đã', 'nói', 'rằng', '&quot;', '<unk>', '&quot;', '<unk>']
INPUT ['And', 'I', 'was', 'very', 'proud', '.']
REF ['Tôi', 'đã', 'rất', 'tự', 'hào', 'về', 'đất', 'nước', 'tôi', '.']
MT ['Và', 'tôi', 'đã', 'nói', 'về', 'một', 'điều', 'đó', '.']
INPUT ['In', 'school', ',', 'we', 'spent', 'a', 'lot', 'of', 'time', 'studying', 'the', 'history', 'of', 'Kim', 'Il-Sung', ',',
'but', 'we', 'never', 'learned', 'much', 'about', 'the', 'outside', 'world', ',', 'except', 'that', 'America', ',', 'South', 'Kor
ea', ',', 'Japan', 'are', 'the', 'enemies', '.']
REF ['Ở', 'trường', ',', 'chúng', 'tôi', 'dành', 'rất', 'nhiều', 'thời', 'gian', 'để', 'học', 'về', 'cuộc', 'đời', 'của', 'chủ',
'tịch', 'Kim', 'II-', 'Sung', ',', 'nhưng', 'lại', 'không', 'học', 'nhiều', 'về', 'thế', 'giới', 'bên', 'ngoài', ',', 'ngoại', 't
rừ', 'việc', 'Hoa', 'Kỳ', ',', 'Hàn', 'Quốc', 'và', 'Nhật', 'Bản', 'là', 'kẻ', 'thù', 'của', 'chúng', 'tôi', '.']
MT ['Trong', 'một', 'năm', 'trước', ',', 'chúng', 'tôi', 'đã', 'có', 'một', 'vài', 'người', 'khác', 'nhau', ',', 'những', 'ngườ
i', 'đã', 'có', 'thể', 'làm', 'được', 'những', 'người', 'khác', 'nhau', ',', 'và', 'họ', 'đã']
```

```
INPUT ['Thank', 'you', 'very', 'much', '.']
REF ['Cảm', 'ơn', 'rất', 'nhiều', '.']
MT ['Cảm', 'ơn', 'các', 'bạn', 'rất', 'nhiều', '.']
total: 1268
bleu: 0.021606175091362874
```