Lab5_Conditional Sequence-to-Sequence VAE

機器人碩 309605014 王麒睿

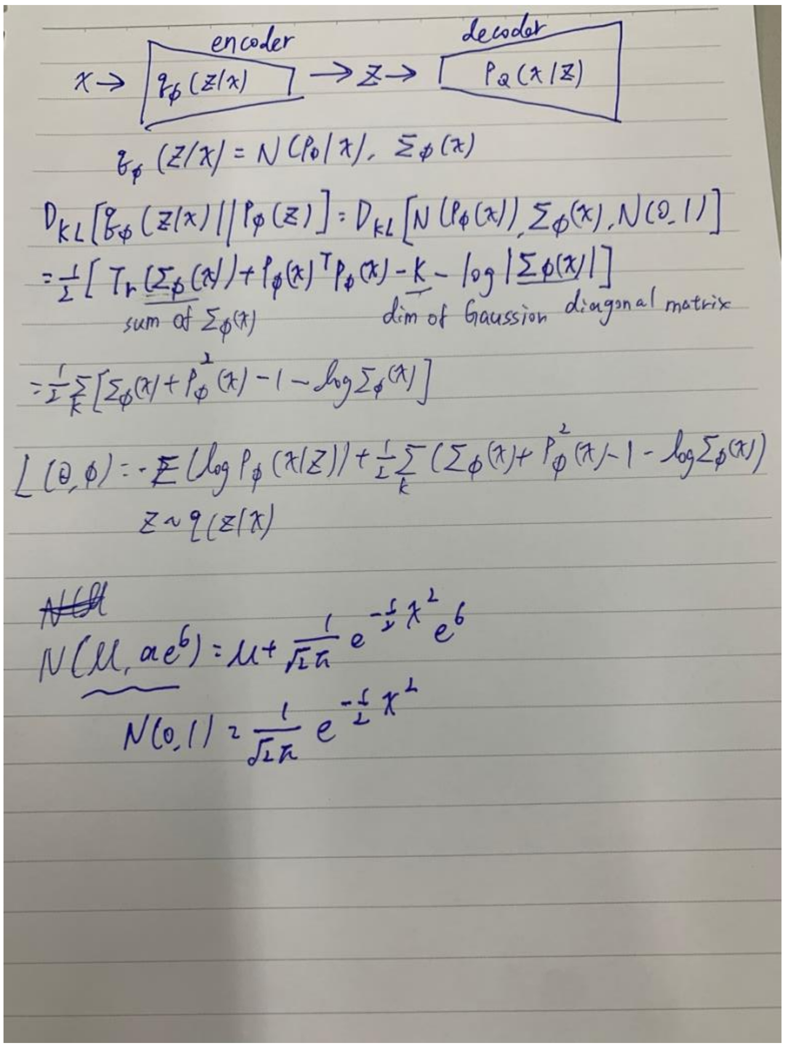
1Introduction

Every English verb has its tenses like simple、simple past etc. To convert different tense between input words and target words, this lab use tense as condition and English words as input and target.

Some requirements as follows:

- Implement a C seq2seq VAE model.
- Adopt two method which teacher-forcing and kl loss annealing to train model.
- Plot cross-entropy loss and kl loss curves during training.
- Plot the BLEU-4 score curve of the test data during training
- Show the results generated words with 4 tenses by Gaussian normal distribution.

2.Derivation of CVAE



$$\begin{tikzcd} x \arrow[r] & \boxed{q_\phi(z|x)} \arrow[r] & z \arrow[r] & \boxed{p_\theta(x|z)} \end{tikzcd}$$

encoder
decoder

$$q_\phi(z|x) = N(\mu_\phi(x), \Sigma_\phi(x))$$

$$D_{KL}[q_\phi(z|x) \| p_\phi(z)] = D_{KL}[N(\mu_\phi(x)), \Sigma_\phi(x), N(0,1)]$$

$$= \frac{1}{2}[\underbrace{T_r(\Sigma_\phi(x))}_{\text{sum of } \Sigma_\phi(x)} + \mu_\phi(x)^T\mu_\phi(x) - \underbrace{k}_{\text{dim of Gaussion}} - \log|\underbrace{\Sigma_\phi(x)}_{\text{diagonal matrix}}|]$$

$$= \frac{1}{2}\sum_k[\Sigma_\phi(x) + \mu_\phi^2(x) - 1 - \log\Sigma_\phi(x)]$$

$$L(\theta,\phi) = -\underset{z \sim q(z|x)}{E}(\log p_\phi(x|z)) + \frac{1}{2}\sum_k(\Sigma_\phi(x) + \mu_\phi^2(x) - 1 - \log\Sigma_\phi(x))$$

~~N(M~~

$$N(\mu, \alpha e^6) = \mu + \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} e^6$$

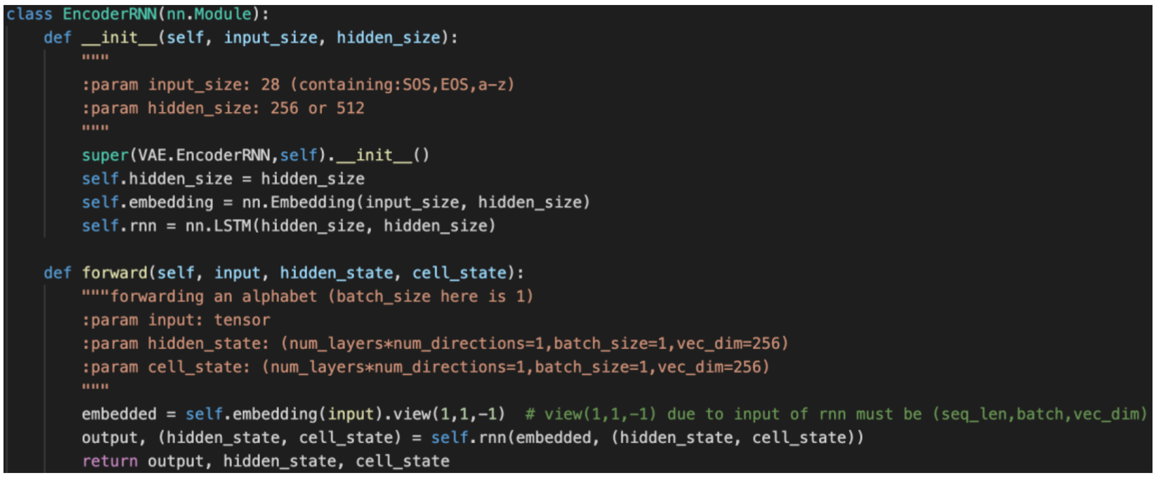
$$N(0,1) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

3.Experimental Setup

CVAE 是由三個部份所組成:Encoder+中間 sample part+Decoder。

Encoder:

hidden_state 加入會先把 alphabet embedding 成一個向量，再丟進 LSTM 去跑，最後輸出 output,hidden_state,cell_state

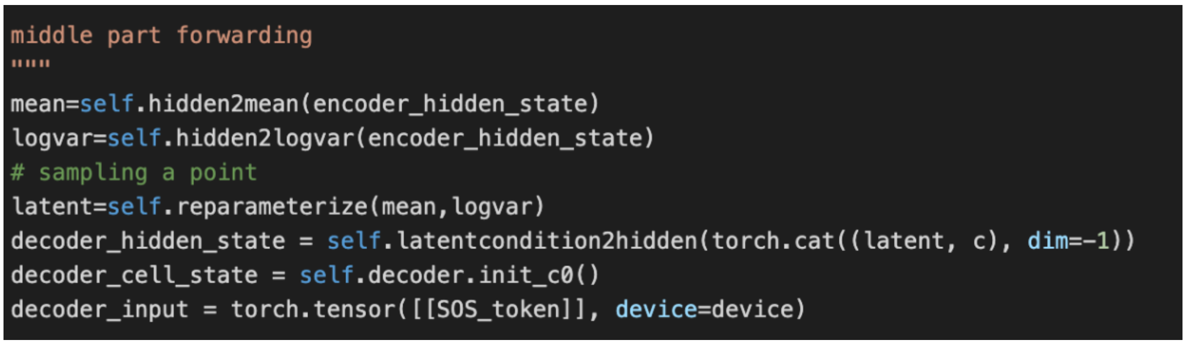


```python
class EncoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        """
        :param input_size: 28 (containing:SOS,EOS,a-z)
        :param hidden_size: 256 or 512
        """
        super(VAE.EncoderRNN,self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(input_size, hidden_size)
        self.rnn = nn.LSTM(hidden_size, hidden_size)

    def forward(self, input, hidden_state, cell_state):
        """forwarding an alphabet (batch_size here is 1)
        :param input: tensor
        :param hidden_state: (num_layers*num_directions=1,batch_size=1,vec_dim=256)
        :param cell_state: (num_layers*num_directions=1,batch_size=1,vec_dim=256)
        """
        embedded = self.embedding(input).view(1,1,-1)  # view(1,1,-1) due to input of rnn must be (seq_len,batch,vec_dim)
        output, (hidden_state, cell_state) = self.rnn(embedded, (hidden_state, cell_state))
        return output, hidden_state, cell_state
```

中間 sample part:

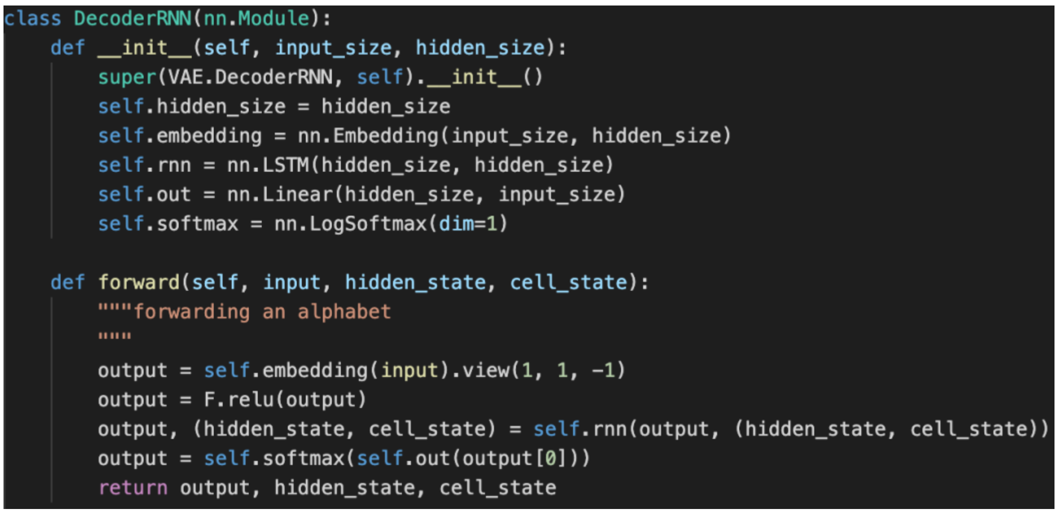
我們把 encoder 輸出的 hidden_state 透過 fully connected layer 變為 32-dim 的 mean 與 log variance，知所以用 log variance 是因為 variance 皆為正值，但 fully connected layer 可能會輸出負數。

有了 mean 與 log variance 後，我們就可以透過 reparameterization trick sampling 一個 32-dim 的 latent，32-dim latent 與 8-dim condition concate 後，在透過一個 fully connected layer 轉換為 hidden_state 的維度。



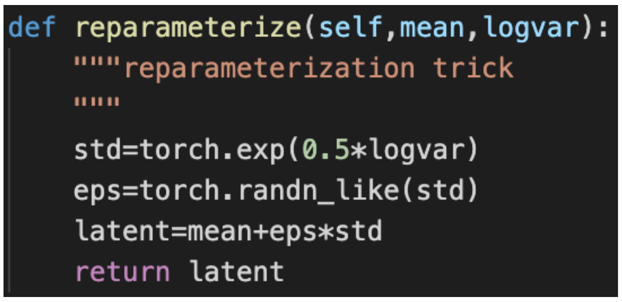
```python
middle part forwarding
"""
mean=self.hidden2mean(encoder_hidden_state)
logvar=self.hidden2logvar(encoder_hidden_state)
# sampling a point
latent=self.reparameterize(mean,logvar)
decoder_hidden_state = self.latentcondition2hidden(torch.cat((latent, c), dim=-1))
decoder_cell_state = self.decoder.init_c0()
decoder_input = torch.tensor([[SOS_token]], device=device)
```

Decoder:
輸入的 hidden_state 為剛剛”中間 sample part”的輸出，cell_state 則初始化為 0 tensor。



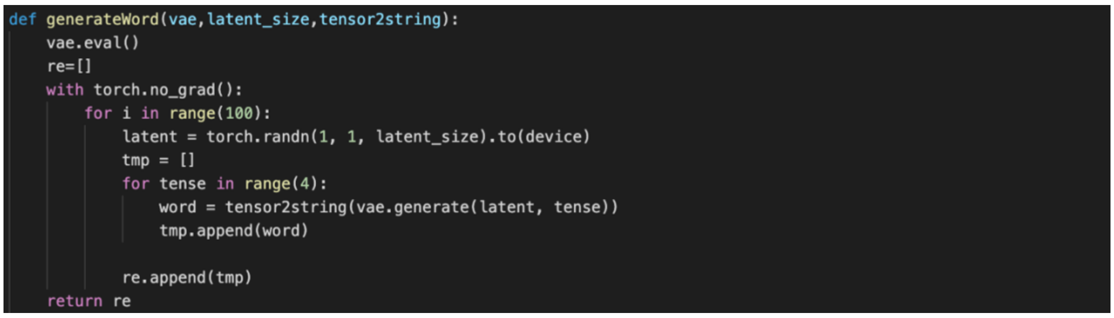
```python
class DecoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(VAE.DecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.embedding = nn.Embedding(input_size, hidden_size)
        self.rnn = nn.LSTM(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, input_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden_state, cell_state):
        """forwarding an alphabet
        """
        output = self.embedding(input).view(1, 1, -1)
        output = F.relu(output)
        output, (hidden_state, cell_state) = self.rnn(output, (hidden_state, cell_state))
        output = self.softmax(self.out(output[0]))
        return output, hidden_state, cell_state
```

Reparameterization trick:
從 N(mean,exp(log variane))中 sample 一個點

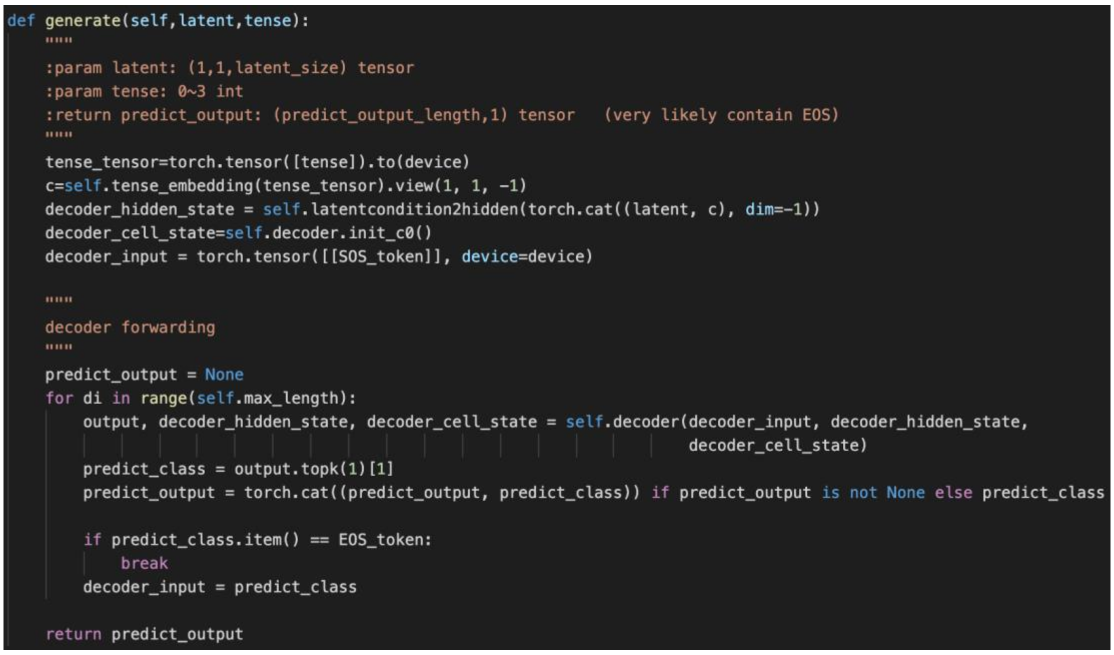


```python
def reparameterize(self,mean,logvar):
    """reparameterization trick
    """
    std=torch.exp(0.5*logvar)
    eps=torch.randn_like(std)
    latent=mean+eps*std
    return latent
```

Text generation by Gaussian noise:
用 torch.randn()隨機生成一個 32-dim 的 latent tensor，再把這 latent tensor 與 tense tensor concate，並作為 decoder 的 hidden_state



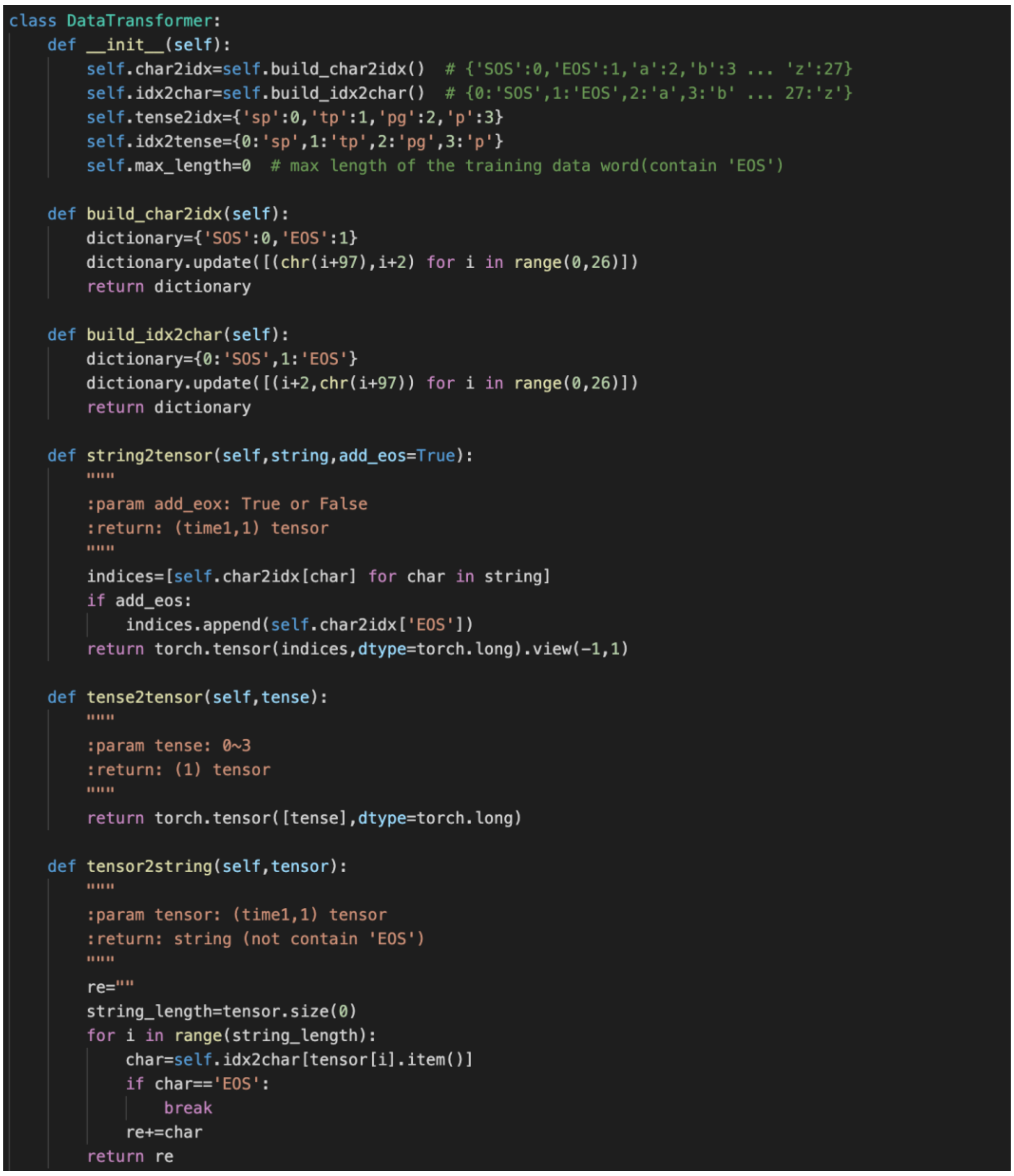
```python
def generateWord(vae,latent_size,tensor2string):
    vae.eval()
    re=[]
    with torch.no_grad():
        for i in range(100):
            latent = torch.randn(1, 1, latent_size).to(device)
            tmp = []
            for tense in range(4):
                word = tensor2string(vae.generate(latent, tense))
                tmp.append(word)

            re.append(tmp)
    return re
```



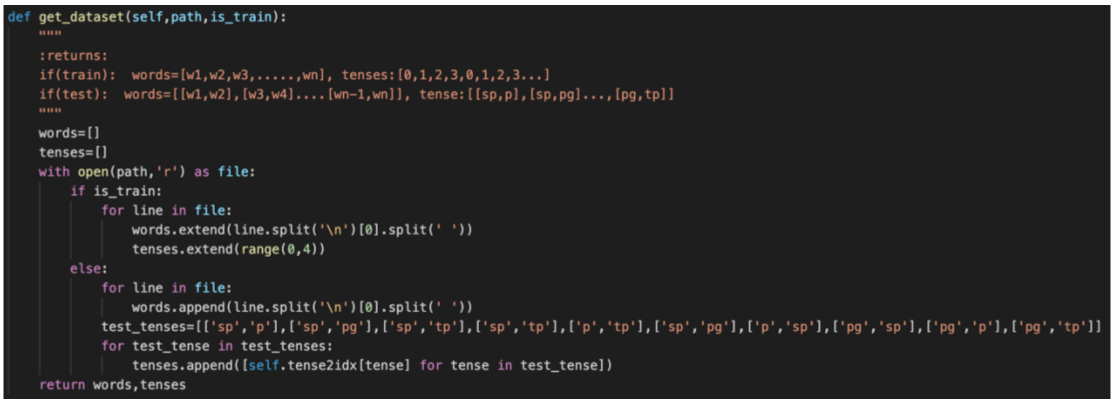
```python
def generate(self,latent,tense):
    """
    :param latent: (1,1,latent_size) tensor
    :param tense: 0~3 int
    :return predict_output: (predict_output_length,1) tensor   (very likely contain EOS)
    """
    tense_tensor=torch.tensor([tense]).to(device)
    c=self.tense_embedding(tense_tensor).view(1, 1, -1)
    decoder_hidden_state = self.latentcondition2hidden(torch.cat((latent, c), dim=-1))
    decoder_cell_state=self.decoder.init_c0()
    decoder_input = torch.tensor([[SOS_token]], device=device)

    """
    decoder forwarding
    """
    predict_output = None
    for di in range(self.max_length):
        output, decoder_hidden_state, decoder_cell_state = self.decoder(decoder_input, decoder_hidden_state,
                                                                        decoder_cell_state)
        predict_class = output.topk(1)[1]
        predict_output = torch.cat((predict_output, predict_class)) if predict_output is not None else predict_class

        if predict_class.item() == EOS_token:
            break
        decoder_input = predict_class

    return predict_output
```

dataloader:

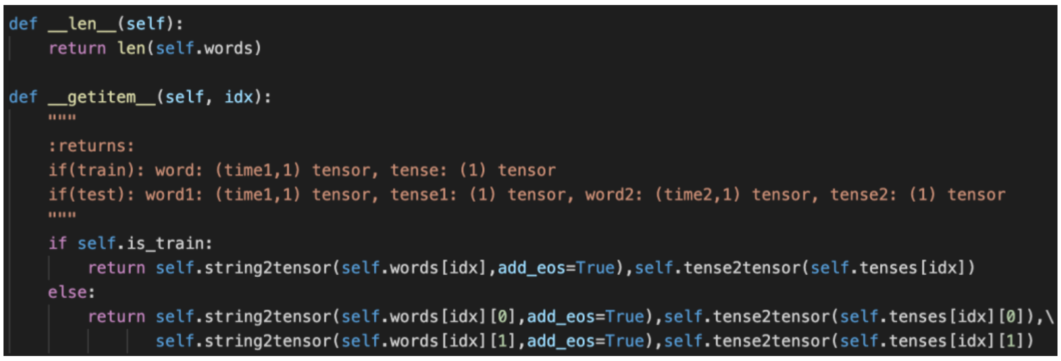
把 SOS,EOS,a,b,c,....,z 分別對應到 0~27，以利將來 torch.nn.embedding()



```python
class DataTransformer:
    def __init__(self):
        self.char2idx=self.build_char2idx()  # {'SOS':0,'EOS':1,'a':2,'b':3 ... 'z':27}
        self.idx2char=self.build_idx2char()  # {0:'SOS',1:'EOS',2:'a',3:'b' ... 27:'z'}
        self.tense2idx={'sp':0,'tp':1,'pg':2,'p':3}
        self.idx2tense={0:'sp',1:'tp',2:'pg',3:'p'}
        self.max_length=0  # max length of the training data word(contain 'EOS')

    def build_char2idx(self):
        dictionary={'SOS':0,'EOS':1}
        dictionary.update([(chr(i+97),i+2) for i in range(0,26)])
        return dictionary

    def build_idx2char(self):
        dictionary={0:'SOS',1:'EOS'}
        dictionary.update([(i+2,chr(i+97)) for i in range(0,26)])
        return dictionary

    def string2tensor(self,string,add_eos=True):
        """
        :param add_eox: True or False
        :return: (time1,1) tensor
        """
        indices=[self.char2idx[char] for char in string]
        if add_eos:
            indices.append(self.char2idx['EOS'])
        return torch.tensor(indices,dtype=torch.long).view(-1,1)

    def tense2tensor(self,tense):
        """
        :param tense: 0~3
        :return: (1) tensor
        """
        return torch.tensor([tense],dtype=torch.long)

    def tensor2string(self,tensor):
        """
        :param tensor: (time1,1) tensor
        :return: string (not contain 'EOS')
        """
        re=""
        string_length=tensor.size(0)
        for i in range(string_length):
            char=self.idx2char[tensor[i].item()]
            if char=='EOS':
                break
            re+=char
        return re
```



```python
def get_dataset(self,path,is_train):
    """
    :returns:
    if(train):  words=[w1,w2,w3,.....,wn], tenses:[0,1,2,3,0,1,2,3...]
    if(test):  words=[[w1,w2],[w3,w4]....[wn-1,wn]], tense:[[sp,p],[sp,pg]...,[pg,tp]]
    """
    words=[]
    tenses=[]
    with open(path,'r') as file:
        if is_train:
            for line in file:
                words.extend(line.split('\n')[0].split(' '))
                tenses.extend(range(0,4))
        else:
            for line in file:
                words.append(line.split('\n')[0].split(' '))
            test_tenses=[['sp','p'],['sp','pg'],['sp','tp'],['sp','tp'],['p','tp'],['sp','pg'],['p','sp'],['pg','sp'],['pg','p'],['pg','tp']]
            for test_tense in test_tenses:
                tenses.append([self.tense2idx[tense] for tense in test_tense])
    return words,tenses
```

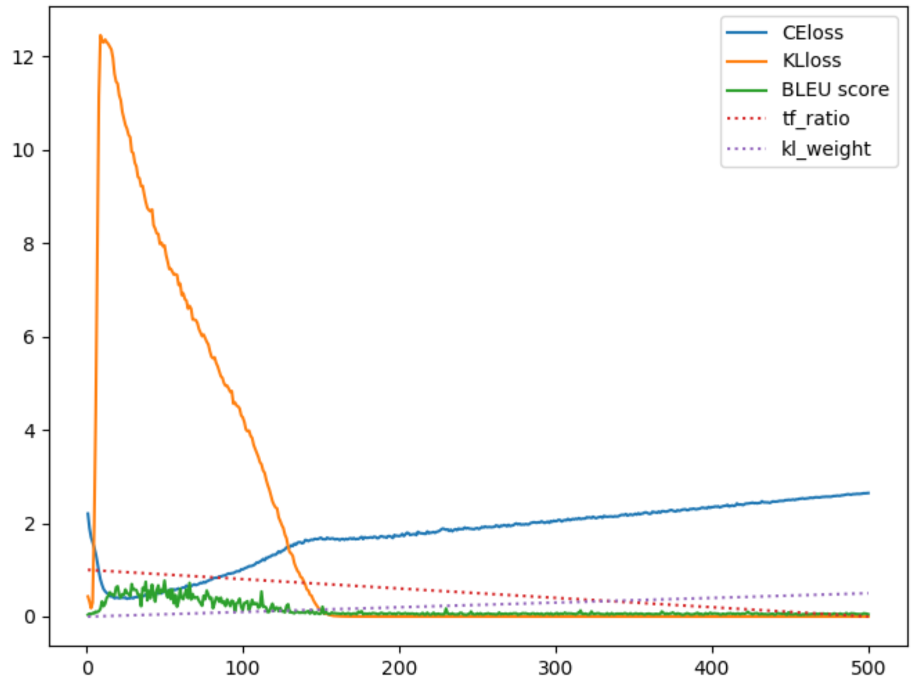


```
def __len__(self):
    return len(self.words)

def __getitem__(self, idx):
    """
    :returns:
    if(train): word: (time1,1) tensor, tense: (1) tensor
    if(test): word1: (time1,1) tensor, tense1: (1) tensor, word2: (time2,1) tensor, tense2: (1) tensor
    """
    if self.is_train:
        return self.string2tensor(self.words[idx],add_eos=True),self.tense2tensor(self.tenses[idx])
    else:
        return self.string2tensor(self.words[idx][0],add_eos=True),self.tense2tensor(self.tenses[idx][0]),\
               self.string2tensor(self.words[idx][1],add_eos=True),self.tense2tensor(self.tenses[idx][1])
```

4.Experimental results

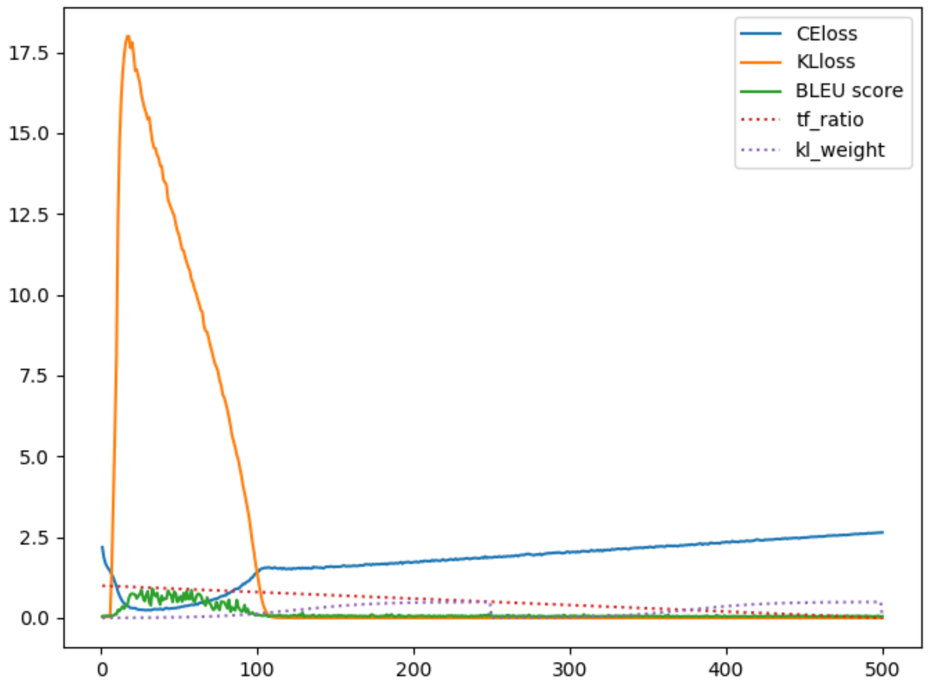
monotonic shedule:

500 個 epoch，teacher_forcing_ratio 從 1.0 線性降至 0.0，kl_weight 從 0.0 線性升至 1.0

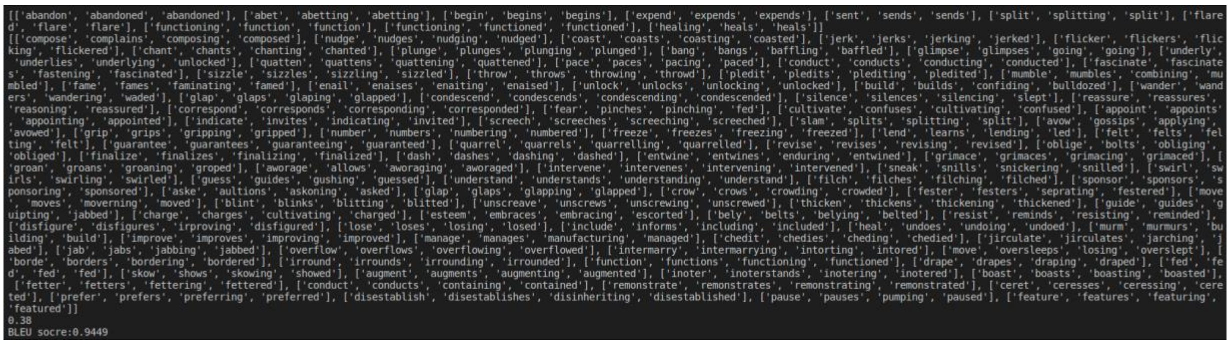


cyclical schedule:

500 個 epoch，teacher_forcing_ratio 從 1.0 線性降至 0.0， 第 1 到 250 個 epoch 間與第 251 到 500 個 epoch 間 kl_weight 從 0.0 線性升至 0.5



Best score:



```
[['abandon', 'abandoned', 'abandoned'], ['abet', 'abetting', 'abetting'], ['begin', 'begins', 'begins'], ['expend', 'expends', 'expends'], ['sent', 'sends', 'sends'], ['split', 'splitting', 'split'], ['flare
d', 'flare', 'flare'], ['functioning', 'function', 'function'], ['functioning', 'functioned', 'functioned'], ['healing', 'heals', 'heals']]
[['compose', 'complains', 'composing', 'composed'], ['nudge', 'nudges', 'nudging', 'nudged'], ['coast', 'coasts', 'coasting', 'coasted'], ['jerk', 'jerks', 'jerking', 'jerked'], ['flicker', 'flickers', 'flic
king', 'flickered'], ['chant', 'chants', 'chanting', 'chanted'], ['plunge', 'plunges', 'plunging', 'plunged'], ['bang', 'bangs', 'baffling', 'baffled'], ['glimpse', 'glimpses', 'going', 'going'], ['underly',
 'underlies', 'underlying', 'unlocked'], ['quatten', 'quattens', 'quattening', 'quattened'], ['pace', 'paces', 'pacing', 'paced'], ['conduct', 'conducts', 'conducting', 'conducted'], ['fascinate', 'fascinate
s', 'fastening', 'fascinated'], ['sizzle', 'sizzles', 'sizzling', 'sizzled'], ['throw', 'throws', 'throwing', 'throwd'], ['pledit', 'pledits', 'plediting', 'pledited'], ['mumble', 'mumbles', 'combining', 'mu
mbled'], ['fame', 'fames', 'faminating', 'famed'], ['enail', 'enaises', 'enaiting', 'enaised'], ['unlock', 'unlocks', 'unlocking', 'unlocked'], ['build', 'builds', 'confiding', 'bulldozed'], ['wander', 'wand
ers', 'wandering', 'waded'], ['glap', 'glaps', 'glaping', 'glapped'], ['condescend', 'condescends', 'condescending', 'condescended'], ['silence', 'silences', 'silencing', 'slept'], ['reassure', 'reassures',
'reasoning', 'reassured'], ['correspond', 'corresponds', 'corresponding', 'corresponded'], ['fear', 'pinches', 'pinching', 'fed'], ['cultivate', 'confuses', 'cultivating', 'confused'], ['appoint', 'appoints'
, 'appointing', 'appointed'], ['indicate', 'invites', 'indicating', 'invited'], ['screech', 'screeches', 'screeching', 'screeched'], ['slam', 'splits', 'splitting', 'split'], ['avow', 'gossips', 'applying',
'avowed'], ['grip', 'grips', 'gripping', 'gripped'], ['number', 'numbers', 'numbering', 'numbered'], ['freeze', 'freezes', 'freezing', 'freezed'], ['lend', 'learns', 'lending', 'led'], ['felt', 'felts', 'fel
ting', 'felt'], ['guarantee', 'guarantees', 'guaranteeing', 'guaranteed'], ['quarrel', 'quarrels', 'quarrelling', 'quarrelled'], ['revise', 'revises', 'revising', 'revised'], ['oblige', 'bolts', 'obliging',
'obliged'], ['finalize', 'finalizes', 'finalizing', 'finalized'], ['dash', 'dashes', 'dashing', 'dashed'], ['entwine', 'entwines', 'enduring', 'entwined'], ['grimace', 'grimaces', 'grimacing', 'grimaced'], [
'groan', 'groans', 'groaning', 'groped'], ['aworage', 'allows', 'aworaging', 'aworaged'], ['intervene', 'intervenes', 'intervening', 'intervened'], ['sneak', 'snills', 'snickering', 'snilled'], ['swirl', 'sw
irls', 'swirling', 'swirled'], ['guess', 'guides', 'gushing', 'guessed'], ['understand', 'understands', 'understanding', 'understand'], ['filch', 'filches', 'filching', 'filched'], ['sponsor', 'sponsors', 's
ponsoring', 'sponsored'], ['aske', 'aultions', 'askoning', 'asked'], ['glap', 'glaps', 'glapping', 'glapped'], ['crow', 'crows', 'crowding', 'crowded'], ['fester', 'festers', 'seprating', 'festered'], ['move
', 'moves', 'moverning', 'moved'], ['blint', 'blinks', 'blitting', 'blitted'], ['unscreave', 'unscrews', 'unscrewing', 'unscrewed'], ['thicken', 'thickens', 'thickening', 'thickened'], ['guide', 'guides', 'g
uipting', 'jabbed'], ['charge', 'charges', 'cultivating', 'charged'], ['esteem', 'embraces', 'embracing', 'escorted'], ['bely', 'belts', 'belying', 'belted'], ['resist', 'reminds', 'resisting', 'reminded'],
['disfigure', 'disfigures', 'irproving', 'disfigured'], ['lose', 'loses', 'losing', 'losed'], ['include', 'informs', 'including', 'included'], ['heal', 'undoes', 'undoing', 'undoed'], ['murm', 'murmurs', 'bu
ilding', 'build'], ['improve', 'improves', 'improving', 'improved'], ['manage', 'manages', 'manufacturing', 'managed'], ['chedit', 'chedies', 'cheding', 'chedied'], ['jirculate', 'jirculates', 'jarching', 'j
abed'], ['jab', 'jabs', 'jabbing', 'jabbed'], ['overflow', 'overflows', 'overflowing', 'overflowed'], ['intermarry', 'intermarrying', 'intorting', 'intored'], ['move', 'oversleeps', 'losing', 'overslept'], [
'borde', 'borders', 'bordering', 'bordered'], ['irround', 'irrounds', 'irrounding', 'irrounded'], ['function', 'functions', 'functioning', 'functioned'], ['drape', 'drapes', 'draping', 'draped'], ['fed', 'fe
d', 'fed', 'fed'], ['skow', 'shows', 'skowing', 'showed'], ['augment', 'augments', 'augmenting', 'augmented'], ['inoter', 'inoterstands', 'inotering', 'inotered'], ['boast', 'boasts', 'boasting', 'boasted'],
 ['fetter', 'fetters', 'fettering', 'fettered'], ['conduct', 'conducts', 'containing', 'contained'], ['remonstrate', 'remonstrates', 'remonstrating', 'remonstrated'], ['ceret', 'ceresses', 'ceressing', 'cere
ted'], ['prefer', 'prefers', 'preferring', 'preferred'], ['disestablish', 'disestablishes', 'disinheriting', 'disestablished'], ['pause', 'pauses', 'pumping', 'paused'], ['feature', 'features', 'featuring',
'featured']]
0.38
BLEU socre:0.9449
```

Bleu:0.9449

5.Discussion

一開始 CE loss 大於 KL loss，model 還沒學到任何東西，BLEU 分數也很小。在大約第 6 個 epoch，時 CE loss 逐漸下降，代表 word 的 reconstuction 成功了，因此 BLEU 提昇; 但 latent distribution 與 prior:N(0,1)長得越來越不像，因此 KL loss 上升。

大約到第 20 個 epoch 時，由於 kl_weight 變大，kl_wight*KL loss 開始 dominate 整個 loss function ，所以 KL loss 會被迫往下降，CE loss 因而上升連帶影響 BLEU 下降。 在更後面的 epoch，由於 kl_weigth 一直提高，迫使 KL loss 一直都很低，BLEU 也連帶無法升高，也發現 KL loss 升高時，BLEU 也會升高。

在 cycle shedule 中，第 250 個 epoch 開始 KL looss 與 BLEU 都應該要上升才對，但並沒有，可能是 KL weight 的 cycle 頻率設的不對，或有其他原因。