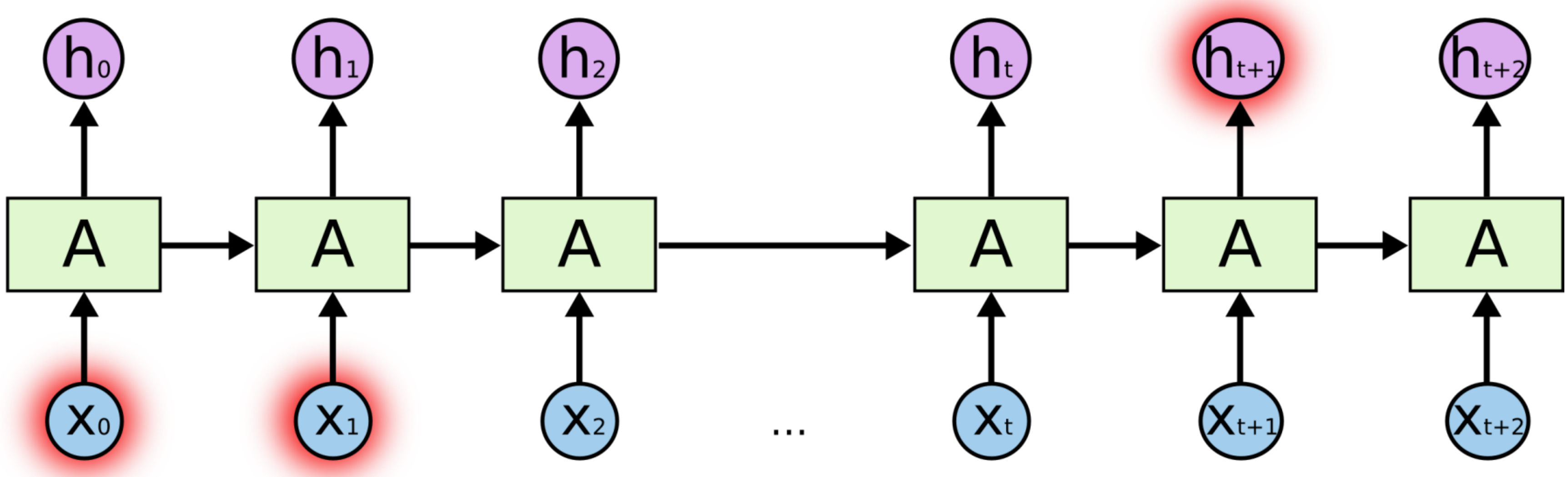
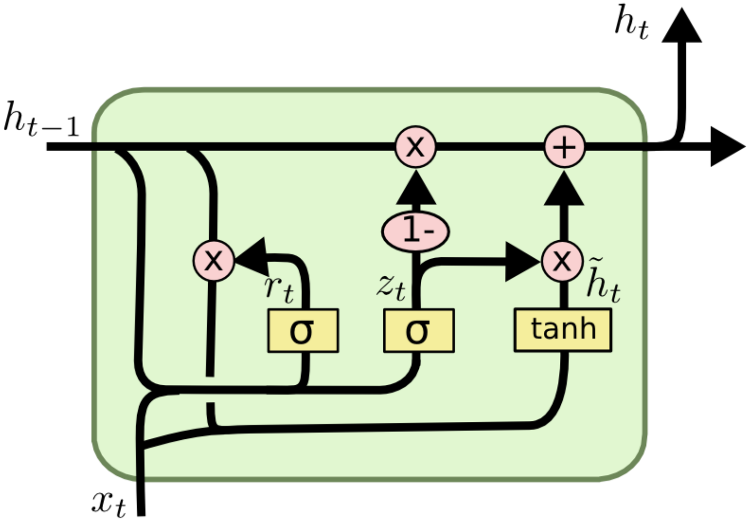
Recurrent Neural Network



LSTM Networks



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

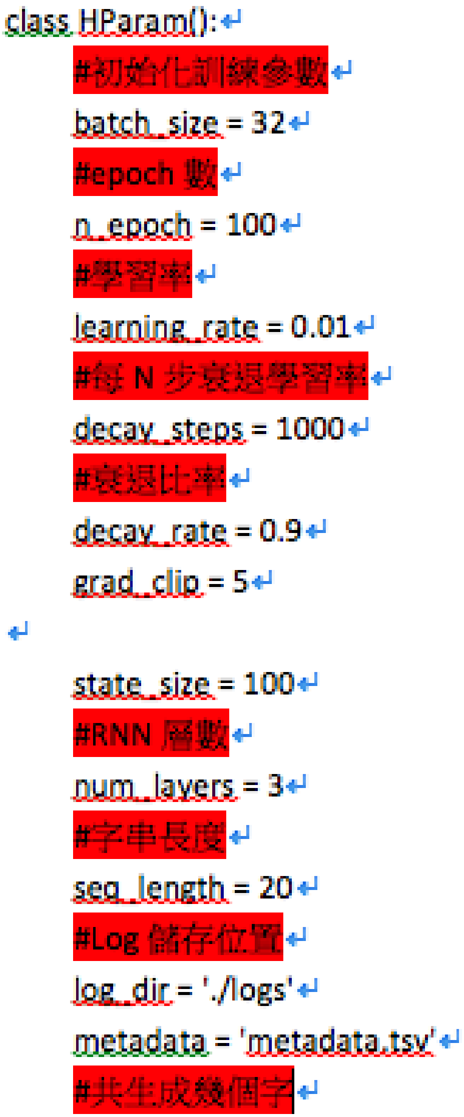
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Task 1 – Character Language Model (CharRNNLM)

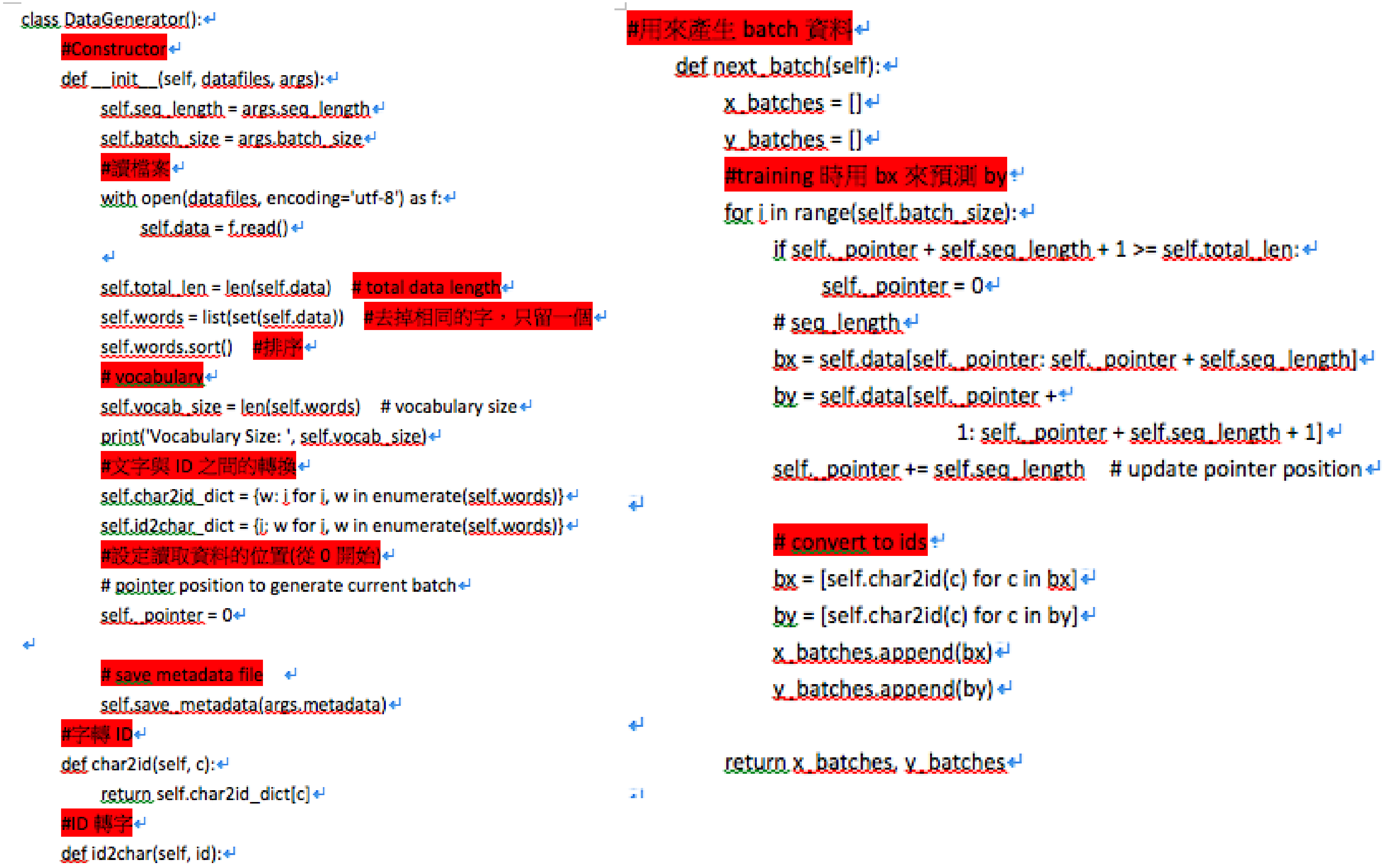
- 實作歌詞生成
  - 要求：任選一個歌手，下載其歌詞
  - 五月天
  - 民歌
  - 兒歌
  - …………
- 範例
  - TensorFlow RNNLM (LSTM)
  - https://www.tensorflow.org/versions/r0.12/tutorials/recurrent/index.html
  - sherjilozair/char-rnn-tensorflow
  - https://github.com/sherjilozair/char-rnn-tensorflow
  - 用TensorFlow生成周杰伦歌词
  - Blog: http://leix.me/2016/11/28/tensorflow-lyrics-generation/
  - Github: https://github.com/leido/char-rnn-cn
  - TensorFlow练习7: 基于RNN生成古诗词
  - http://blog.topspeedsnail.com/archives/10542

Coding Comments-class HParam



```
class HParam():
    #初始化訓練參數
    batch_size = 32
    #epoch 數
    n_epoch = 100
    #學習率
    learning_rate = 0.01
    #每 N 步衰退學習率
    decay_steps = 1000
    #衰退比率
    decay_rate = 0.9
    grad_clip = 5

    state_size = 100
    #RNN 層數
    num_layers = 3
    #字串長度
    seq_length = 20
    #Log 儲存位置
    log_dir = './logs'
    metadata = 'metadata.tsv'
    #共生成幾個字
```

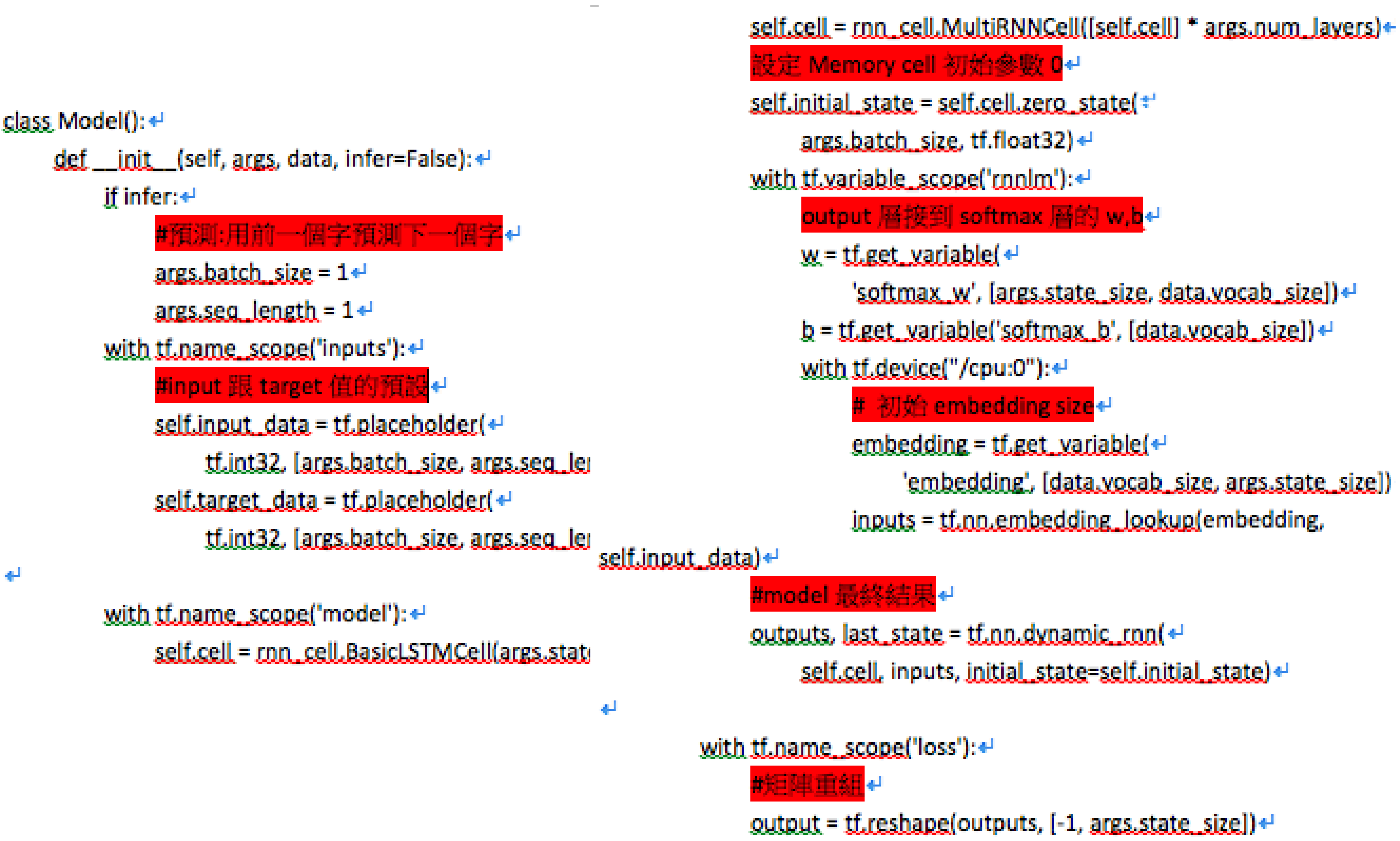
Coding Comments-class DataGenerator



```
class DataGenerator():
    #Constructor
    def __init__(self, datafiles, args):
        self.seq_length = args.seq_length
        self.batch_size = args.batch_size
        #讀檔案
        with open(datafiles, encoding='utf-8') as f:
            self.data = f.read()

        self.total_len = len(self.data)    # total data length
        self.words = list(set(self.data))    #去掉相同的字，只留一個
        self.words.sort()    #排序
        # vocabulary
        self.vocab_size = len(self.words)    # vocabulary size
        print('Vocabulary Size: ', self.vocab_size)
        #文字與 ID 之間的轉換
        self.char2id_dict = {w: i for i, w in enumerate(self.words)}
        self.id2char_dict = {i: w for i, w in enumerate(self.words)}
        #設定讀取資料的位置(從 0 開始)
        # pointer position to generate current batch
        self._pointer = 0


        # save metadata file
        self.save_metadata(args.metadata)
    #字轉 ID
    def char2id(self, c):
        return self.char2id_dict[c]
    #ID 轉字
    def id2char(self, id):
```

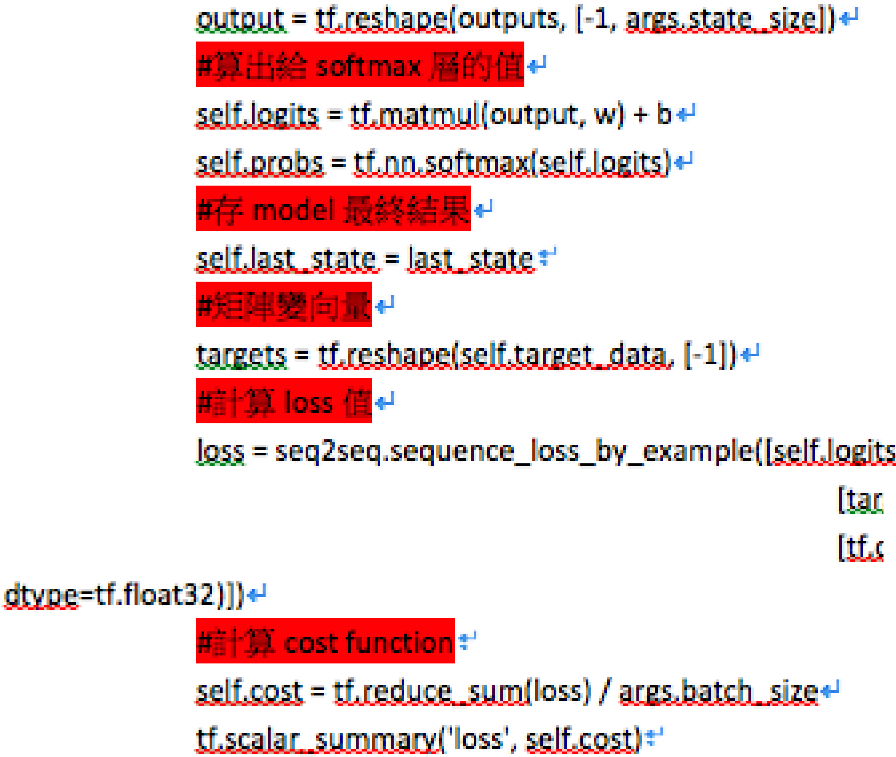
```
#用來產生 batch 資料
    def next_batch(self):
        x_batches = []
        y_batches = []
        #training 時用 bx 來預測 by
        for i in range(self.batch_size):
            if self._pointer + self.seq_length + 1 >= self.total_len:
                self._pointer = 0
            # seq_length
            bx = self.data[self._pointer: self._pointer + self.seq_length]
            by = self.data[self._pointer +
                                1: self._pointer + self.seq_length + 1]
            self._pointer += self.seq_length    # update pointer position


            # convert to ids
            bx = [self.char2id(c) for c in bx]
            by = [self.char2id(c) for c in by]
            x_batches.append(bx)
            y_batches.append(by)


        return x_batches, y_batches
```

Coding Comments-class Model

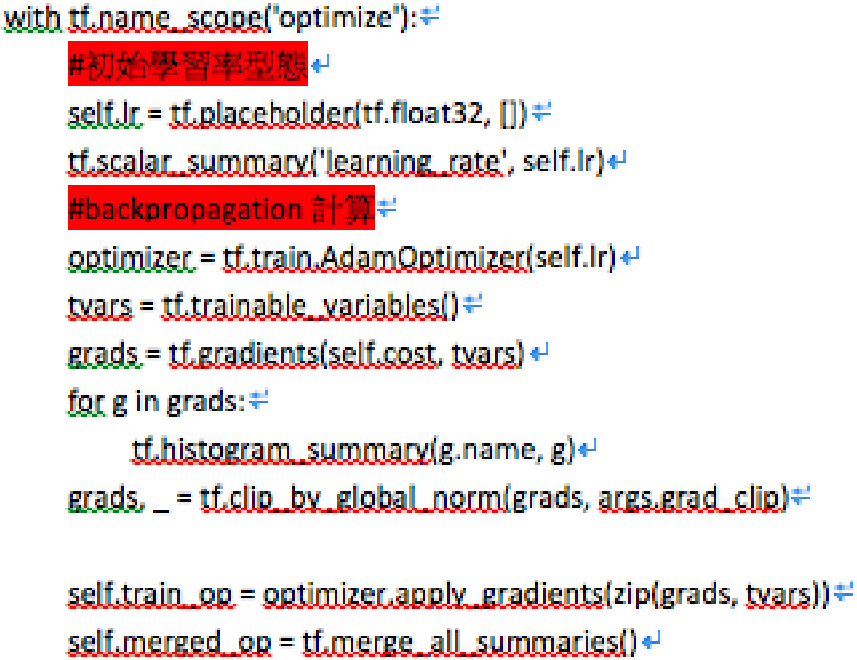


```
class Model():
    def __init__(self, args, data, infer=False):
        if infer:
            #預測:用前一個字預測下一個字
            args.batch_size = 1
            args.seq_length = 1
        with tf.name_scope('inputs'):
            #input 跟 target 值的預設
            self.input_data = tf.placeholder(
                tf.int32, [args.batch_size, args.seq_le
            self.target_data = tf.placeholder(
                tf.int32, [args.batch_size, args.seq_le

        with tf.name_scope('model'):
            self.cell = rnn_cell.BasicLSTMCell(args.stat
            self.cell = rnn_cell.MultiRNNCell([self.cell] * args.num_layers)
            設定 Memory cell 初始參數 0
            self.initial_state = self.cell.zero_state(
                args.batch_size, tf.float32)
            with tf.variable_scope('rnnlm'):
                output 層接到 softmax 層的 w,b
                w = tf.get_variable(
                    'softmax_w', [args.state_size, data.vocab_size])
                b = tf.get_variable('softmax_b', [data.vocab_size])
                with tf.device("/cpu:0"):
                    #  初始 embedding size
                    embedding = tf.get_variable(
                        'embedding', [data.vocab_size, args.state_size])
                    inputs = tf.nn.embedding_lookup(embedding,
self.input_data)
            #model 最終結果
            outputs, last_state = tf.nn.dynamic_rnn(
                self.cell, inputs, initial_state=self.initial_state)

        with tf.name_scope('loss'):
            #矩陣重組
            output = tf.reshape(outputs, [-1, args.state_size])
```

Coding Comments-class Model

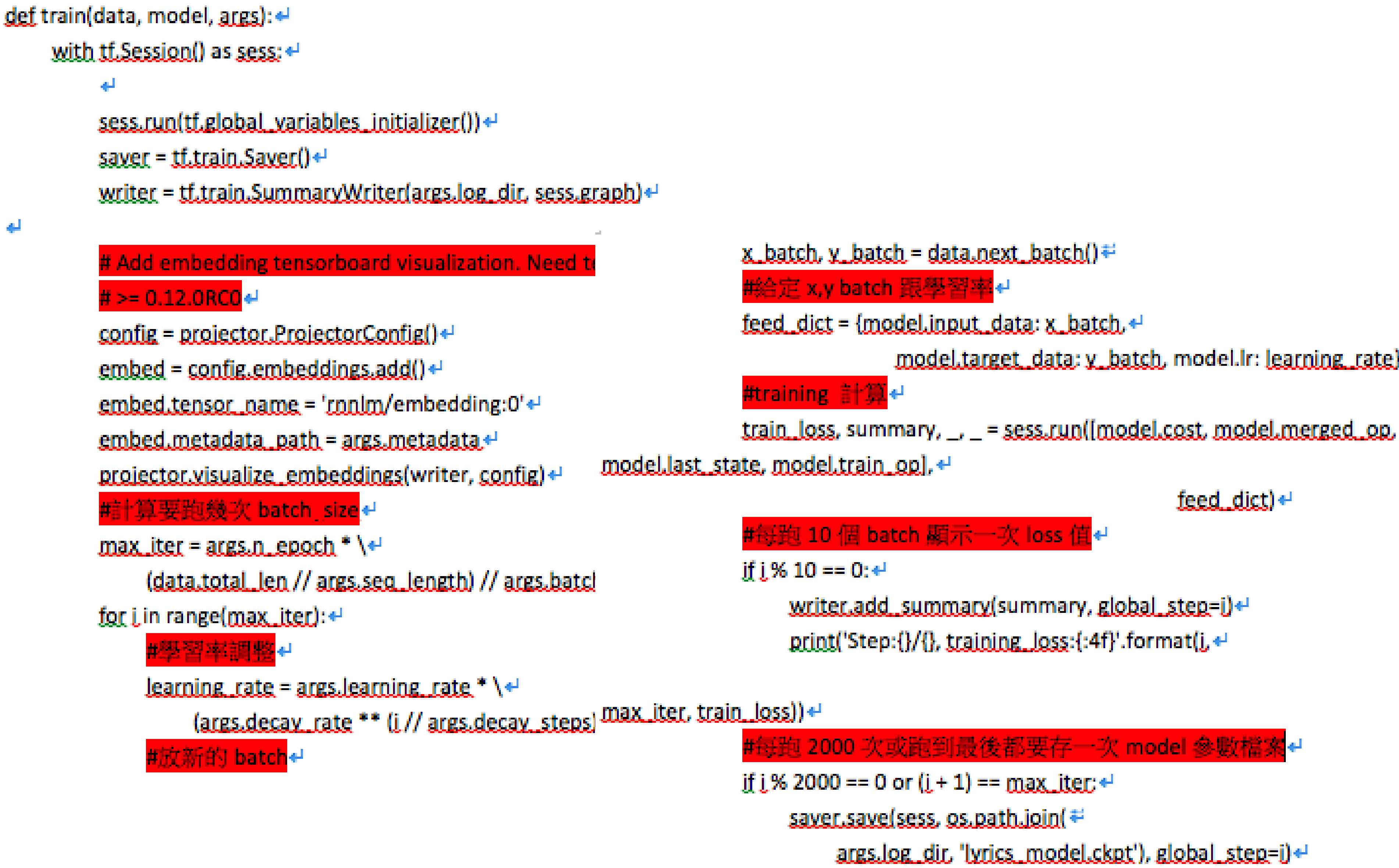


```
output = tf.reshape(outputs, [-1, args.state_size])
#算出給 softmax 層的值
self.logits = tf.matmul(output, w) + b
self.probs = tf.nn.softmax(self.logits)
#存 model 最終結果
self.last_state = last_state
#矩陣變向量
targets = tf.reshape(self.target_data, [-1])
#計算 loss 值
loss = seq2seq.sequence_loss_by_example([self.logits],
                                        [tar
                                        [tf.c
dtype=tf.float32)])
#計算 cost function
self.cost = tf.reduce_sum(loss) / args.batch_size
tf.scalar_summary('loss', self.cost)
```



```
with tf.name_scope('optimize'):
    #初始學習率型態
    self.lr = tf.placeholder(tf.float32, [])
    tf.scalar_summary('learning_rate', self.lr)
    #backpropagation 計算
    optimizer = tf.train.AdamOptimizer(self.lr)
    tvars = tf.trainable_variables()
    grads = tf.gradients(self.cost, tvars)
    for g in grads:
        tf.histogram_summary(g.name, g)
    grads, _ = tf.clip_by_global_norm(grads, args.grad_clip)

    self.train_op = optimizer.apply_gradients(zip(grads, tvars))
    self.merged_op = tf.merge_all_summaries()
```

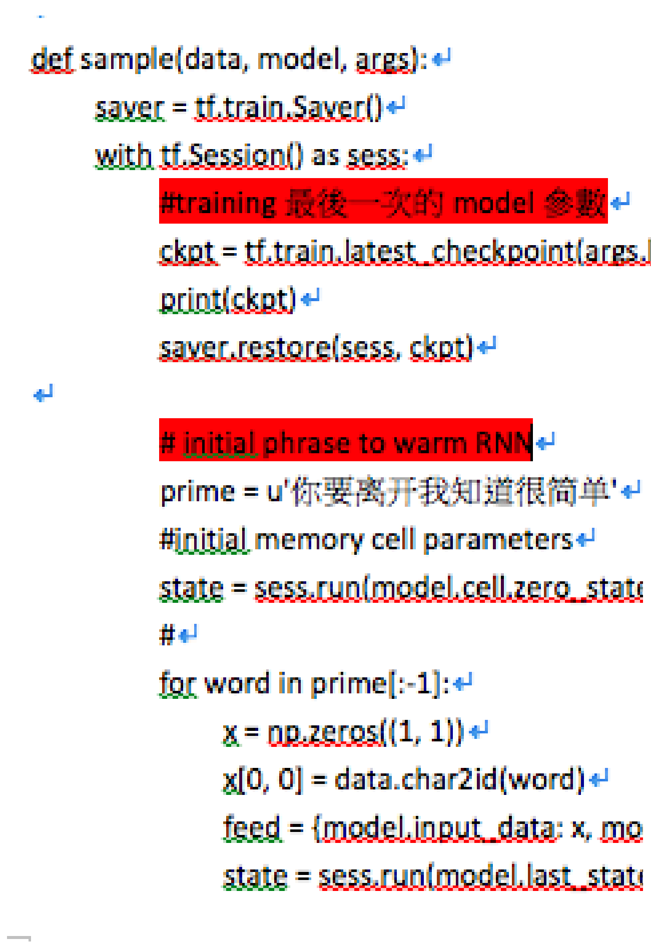
Coding Comments-train function



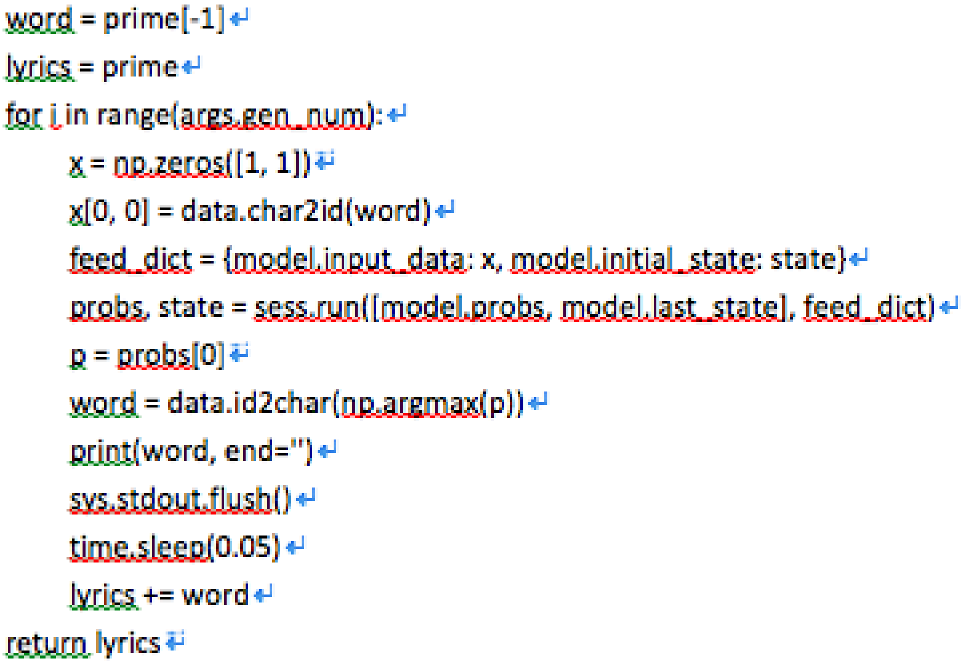
```
def train(data, model, args):
    with tf.Session() as sess:

        sess.run(tf.global_variables_initializer())
        saver = tf.train.Saver()
        writer = tf.train.SummaryWriter(args.log_dir, sess.graph)

        # Add embedding tensorboard visualization. Need t
        # >= 0.12.0RC0
        config = projector.ProjectorConfig()
        embed = config.embeddings.add()
        embed.tensor_name = 'rnnlm/embedding:0'
        embed.metadata_path = args.metadata
        projector.visualize_embeddings(writer, config)
        #計算要跑幾次 batch_size
        max_iter = args.n_epoch * \
            (data.total_len // args.seq_length) // args.batc
        for i in range(max_iter):
            #學習率調整
            learning_rate = args.learning_rate * \
                (args.decay_rate ** (i // args.decay_steps)
            #放新的 batch
```

```
            x_batch, y_batch = data.next_batch()
            #給定 x,y batch 跟學習率
            feed_dict = {model.input_data: x_batch,
                         model.target_data: y_batch, model.lr: learning_rate
            #training 計算
            train_loss, summary, _, _ = sess.run([model.cost, model.merged_op,
model.last_state, model.train_op],
                                                  feed_dict)
            #每跑 10 個 batch 顯示一次 loss 值
            if i % 10 == 0:
                writer.add_summary(summary, global_step=i)
                print('Step:{}/{}, training_loss:{:4f}'.format(i,
max_iter, train_loss))
            #每跑 2000 次或跑到最後都要存一次 model 參數檔案
            if i % 2000 == 0 or (i + 1) == max_iter:
                saver.save(sess, os.path.join(
                    args.log_dir, 'lyrics_model.ckpt'), global_step=i)
```

Coding Comments-sample function

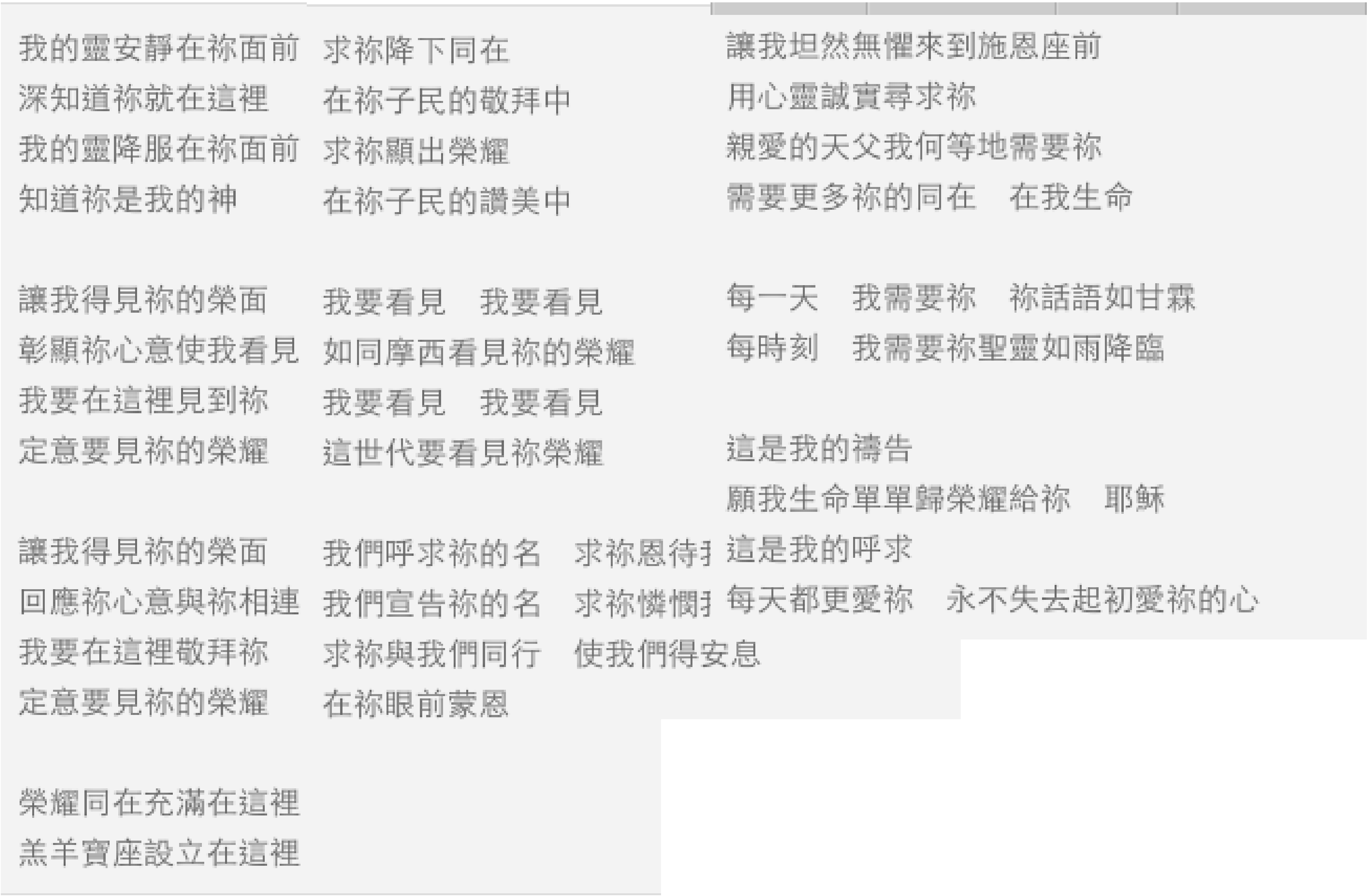


```
def sample(data, model, args):
    saver = tf.train.Saver()
    with tf.Session() as sess:
        #training 最後一次的 model 參數
        ckpt = tf.train.latest_checkpoint(args.log_dir)
        print(ckpt)
        saver.restore(sess, ckpt)

        # initial phrase to warm RNN
        prime = u'你要离开我知道很简单'
        #initial memory cell parameters
        state = sess.run(model.cell.zero_stat
        #
        for word in prime[:-1]:
            x = np.zeros((1, 1))
            x[0, 0] = data.char2id(word)
            feed = {model.input_data: x, mo
            state = sess.run(model.last_stat
```



```
        word = prime[-1]
        lyrics = prime
        for i in range(args.gen_num):
            x = np.zeros([1, 1])
            x[0, 0] = data.char2id(word)
            feed_dict = {model.input_data: x, model.initial_state: state}
            probs, state = sess.run([model.probs, model.last_state], feed_dict)
            p = probs[0]
            word = data.id2char(np.argmax(p))
            print(word, end='')
            sys.stdout.flush()
            time.sleep(0.05)
            lyrics += word
        return lyrics
```

Training data set-讚美之泉



我的靈安靜在祢面前
深知道祢就在這裡
我的靈降服在祢面前
知道祢是我的神

讓我得見祢的榮面
彰顯祢心意使我看見
我要在這裡見到祢
定意要見祢的榮耀

讓我得見祢的榮面
回應祢心意與祢相連
我要在這裡敬拜祢
定意要見祢的榮耀

榮耀同在充滿在這裡
羔羊寶座設立在這裡

求祢降下同在
在祢子民的敬拜中
求祢顯出榮耀
在祢子民的讚美中

我要看見　我要看見
如同摩西看見祢的榮耀
我要看見　我要看見
這世代要看見祢榮耀

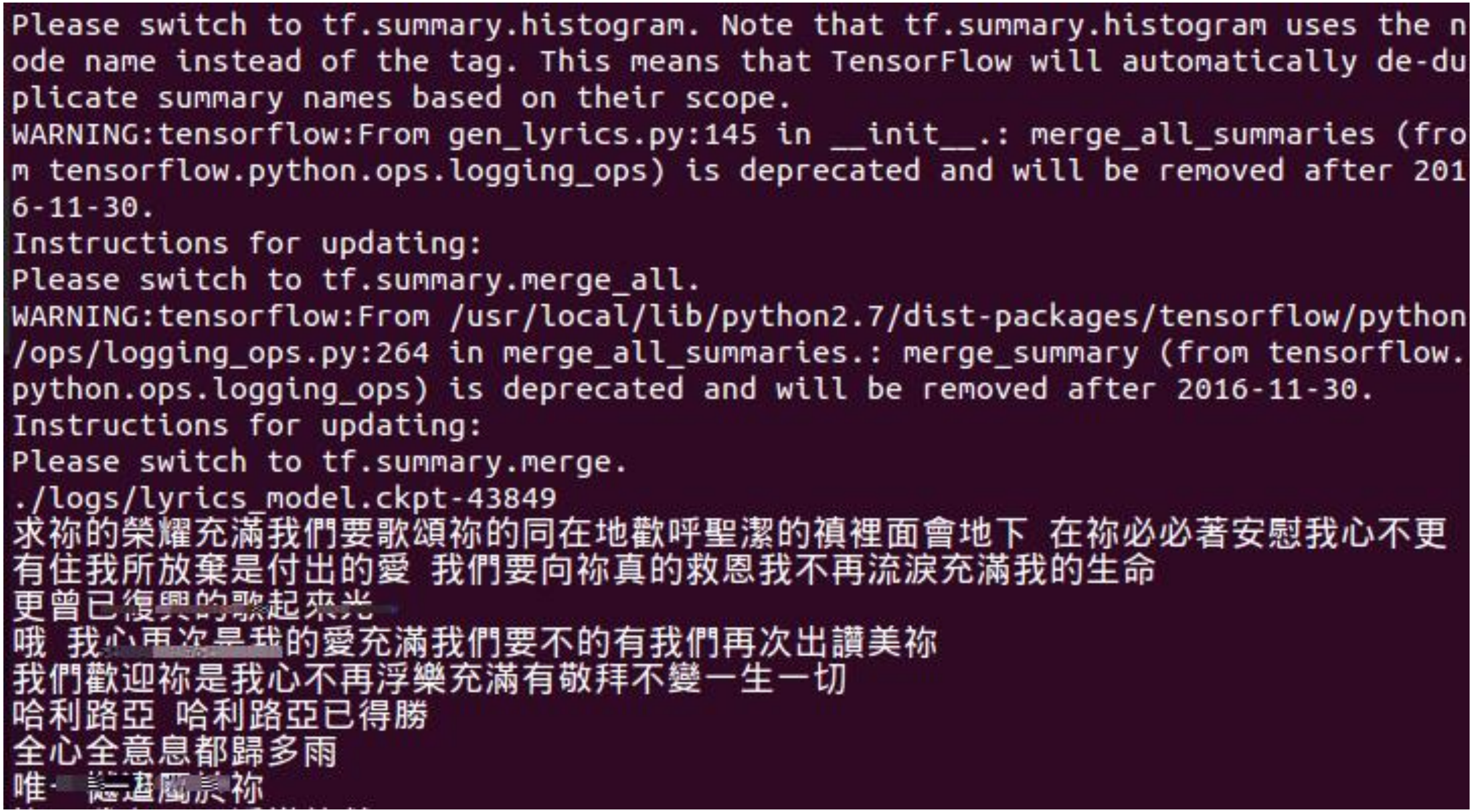
我們呼求祢的名　求祢恩待
我們宣告祢的名　求祢憐憫
求祢與我們同行　使我們得安息
在祢眼前蒙恩

讓我坦然無懼來到施恩座前
用心靈誠實尋求祢
親愛的天父我何等地需要祢
需要更多祢的同在　在我生命

每一天　我需要祢　祢話語如甘霖
每時刻　我需要祢聖靈如雨降臨

這是我的禱告
願我生命單單歸榮耀給祢　耶穌
這是我的呼求
每天都更愛祢　永不失去起初愛祢的心

Result



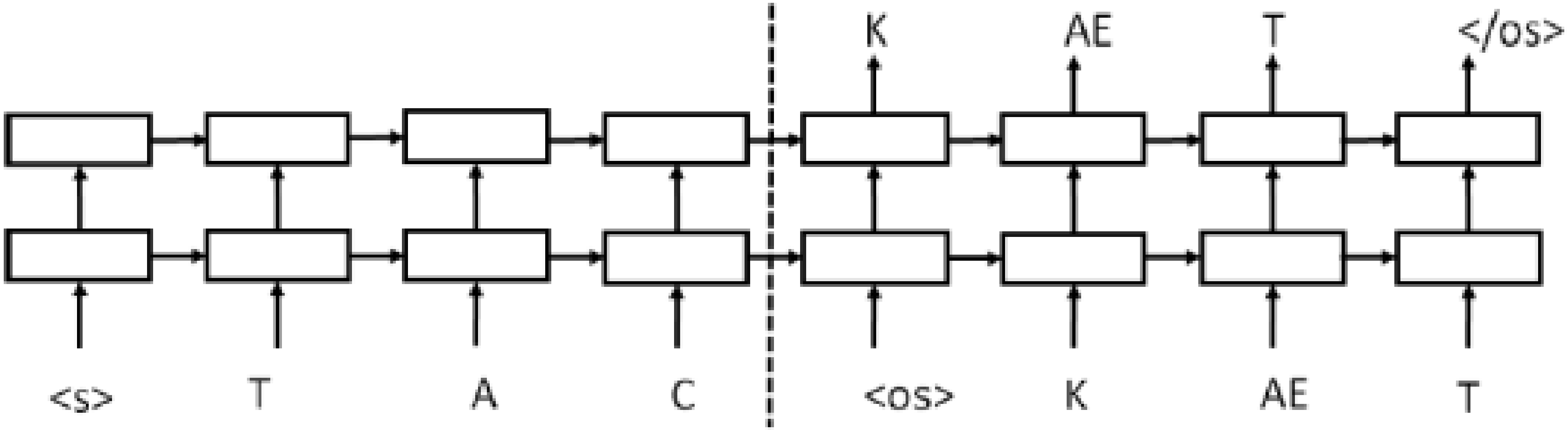
```
Please switch to tf.summary.histogram. Note that tf.summary.histogram uses the n
ode name instead of the tag. This means that TensorFlow will automatically de-du
plicate summary names based on their scope.
WARNING:tensorflow:From gen_lyrics.py:145 in __init__.: merge_all_summaries (fro
m tensorflow.python.ops.logging_ops) is deprecated and will be removed after 201
6-11-30.
Instructions for updating:
Please switch to tf.summary.merge_all.
WARNING:tensorflow:From /usr/local/lib/python2.7/dist-packages/tensorflow/python
/ops/logging_ops.py:264 in merge_all_summaries.: merge_summary (from tensorflow.
python.ops.logging_ops) is deprecated and will be removed after 2016-11-30.
Instructions for updating:
Please switch to tf.summary.merge.
./logs/lyrics_model.ckpt-43849
求祢的榮耀充滿我們要歌頌祢的同在地歡呼聖潔的禎裡面會地下 在祢必必著安慰我心不更
有住我所放棄是付出的愛 我們要向祢真的救恩我不再流淚充滿我的生命
更曾已復興的歌起來光
哦 我心再次是我的愛充滿我們要不的有我們再次出讚美祢
我們歡迎祢是我心不再浮樂充滿有敬拜不變一生一切
哈利路亞 哈利路亞已得勝
全心全意息都歸多雨
唯[illegible]祢
```

Task2 - Grapheme-to-Phoneme (G2P)

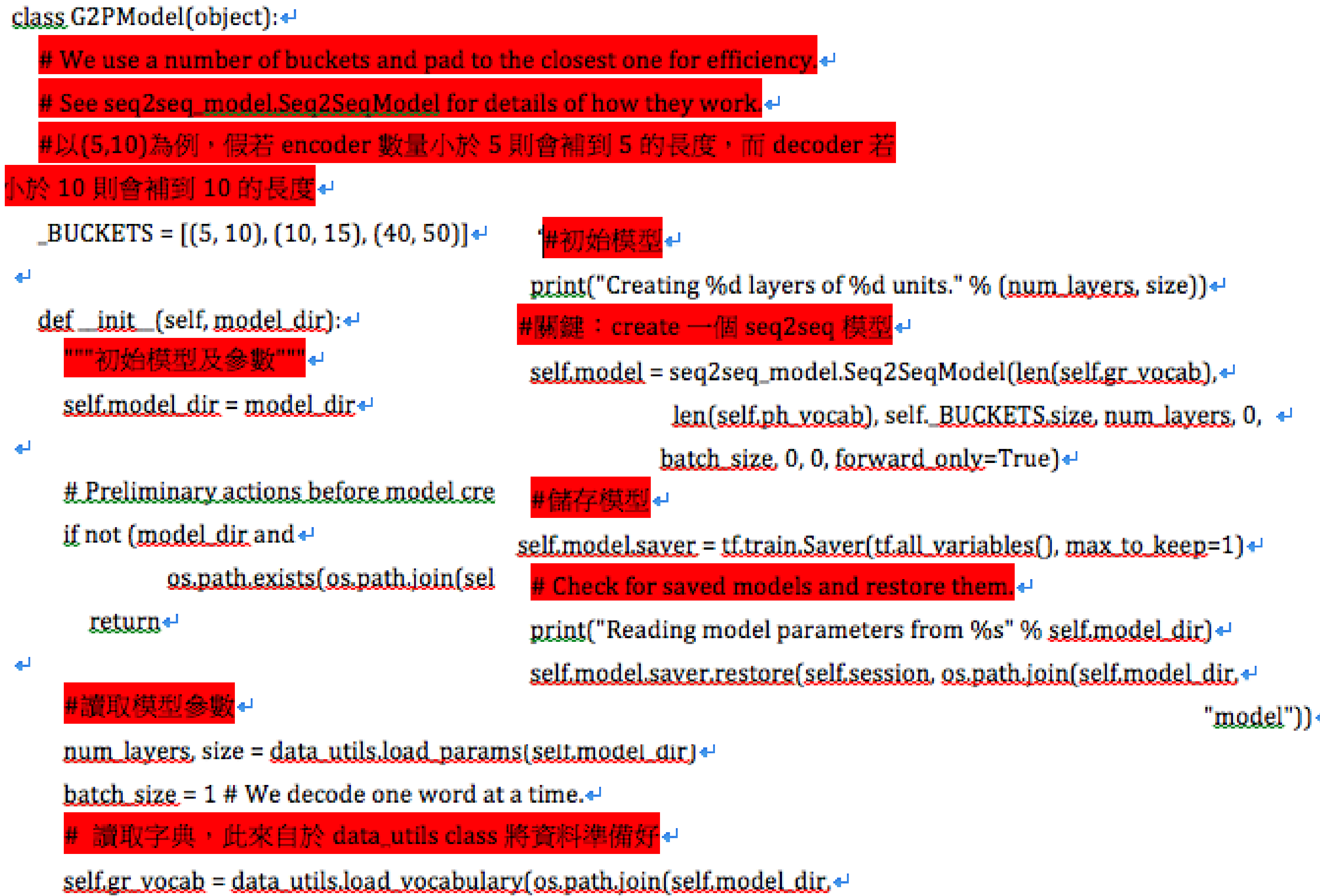
- 實作字轉音（字查發音） or 音轉字（拼音輸入法）
  - 要求：任選一個字典實作
  - 國語
  - 台語
  - 客語
  - …………
- 範例
  - TensorFlow Sequence-to-Sequence Models
  - https://www.tensorflow.org/versions/r0.12/tutorials/seq2seq/index.html
  - Sequence-to-Sequence G2P toolkit
  - https://github.com/cmusphinx/g2p-seq2seq
  - G2P(单词到音素)的深度学习训练测试
  - http://www.itdadao.com/articles/c15a94139p0.html
  - 基于TensorFlow实现的闲聊机器人
  - https://github.com/qhduan/Seq2Seq_Chatbot_QA
  - Reference:

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/rnnlts.pdf

Seq2seq-Model



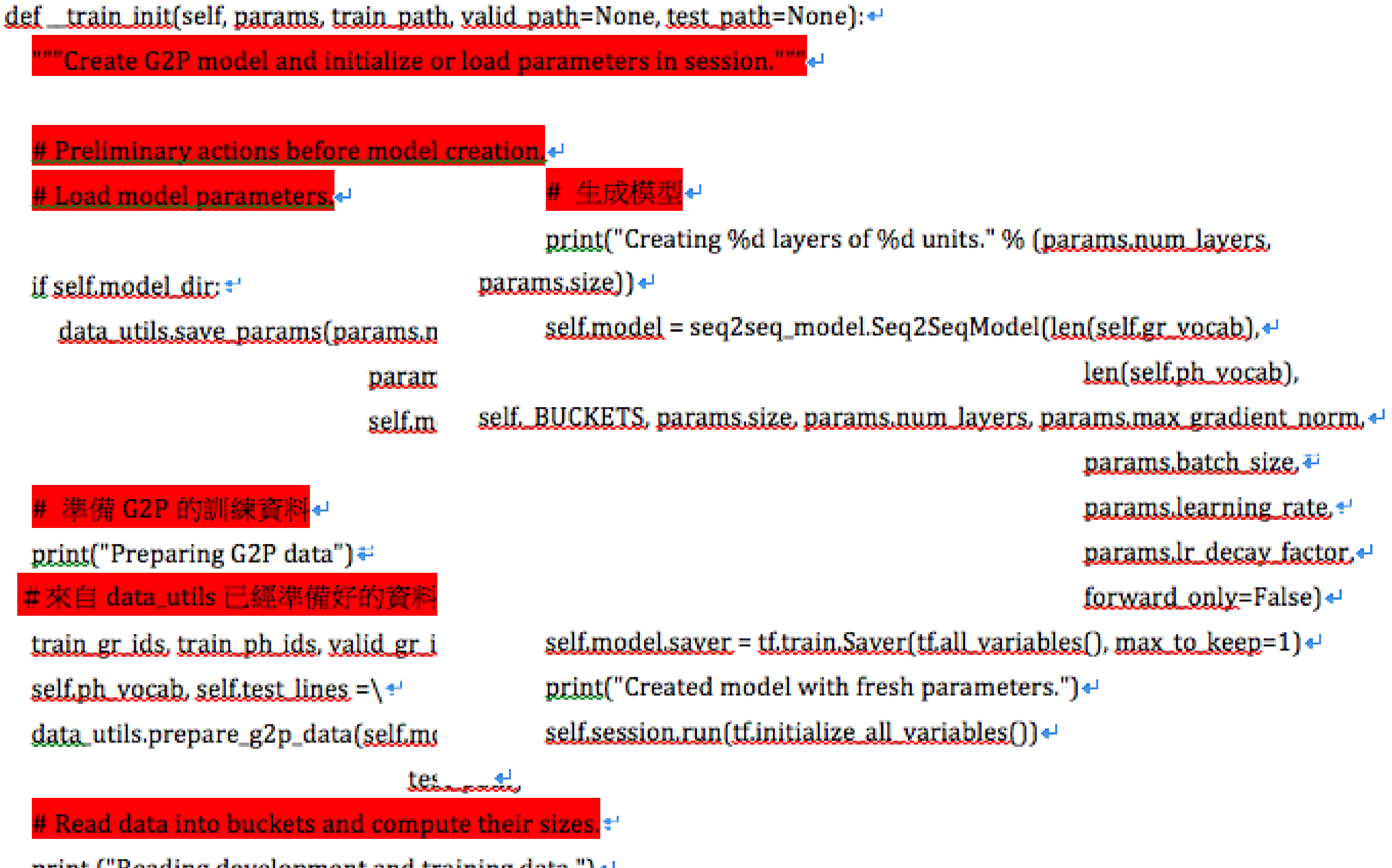
Coding Comments-class G2PModel



```
class G2PModel(object):
  # We use a number of buckets and pad to the closest one for efficiency.
  # See seq2seq_model.Seq2SeqModel for details of how they work.
  #以(5,10)為例，假若 encoder 數量小於 5 則會補到 5 的長度，而 decoder 若
小於 10 則會補到 10 的長度
  _BUCKETS = [(5, 10), (10, 15), (40, 50)]

  def __init__(self, model_dir):
    """初始模型及參數"""
    self.model_dir = model_dir

    # Preliminary actions before model cre
    if not (model_dir and
            os.path.exists(os.path.join(sel
      return

    #讀取模型參數
    num_layers, size = data_utils.load_params(self.model_dir)
    batch_size = 1 # We decode one word at a time.
    #  讀取字典，此來自於 data_utils class 將資料準備好
    self.gr_vocab = data_utils.load_vocabulary(os.path.join(self.model_dir,
```

```
 #初始模型
 print("Creating %d layers of %d units." % (num_layers, size))
#關鍵：create 一個 seq2seq 模型
 self.model = seq2seq_model.Seq2SeqModel(len(self.gr_vocab),
            len(self.ph_vocab), self._BUCKETS,size, num_layers, 0,
           batch_size, 0, 0, forward_only=True)
 #儲存模型
self.model.saver = tf.train.Saver(tf.all_variables(), max_to_keep=1)
 # Check for saved models and restore them.
 print("Reading model parameters from %s" % self.model_dir)
 self.model.saver.restore(self.session, os.path.join(self.model_dir,
                                                      "model"))
```

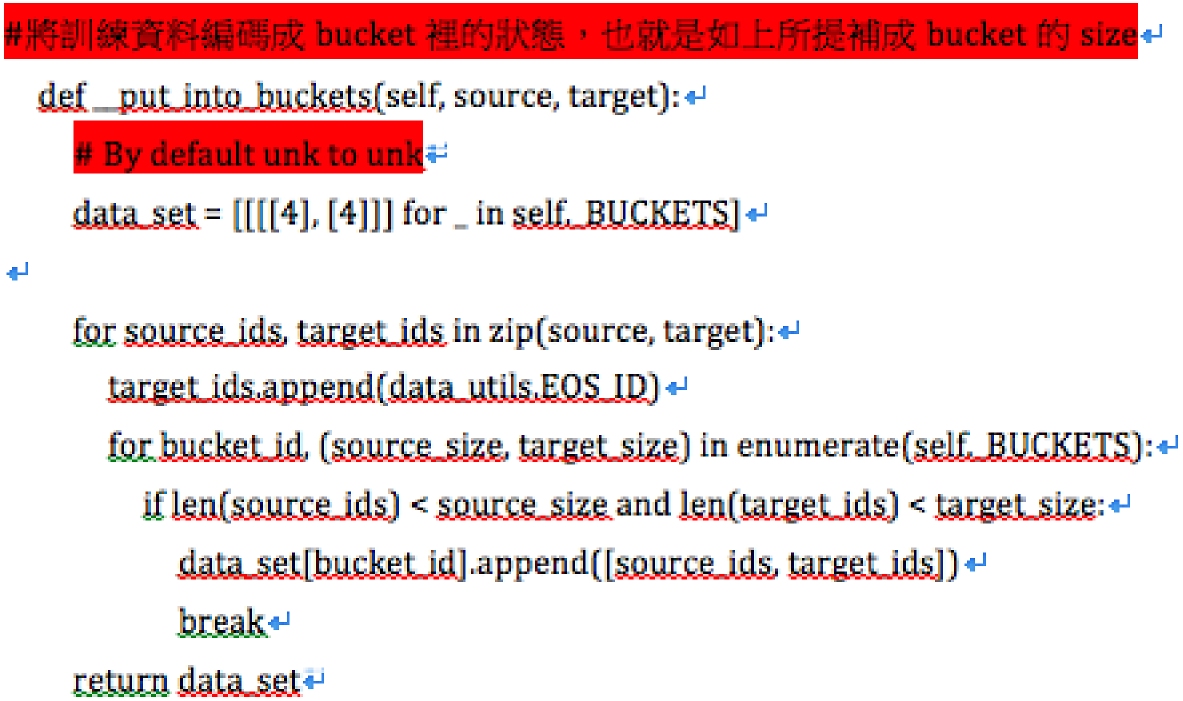
Coding Comments-train init function



```
def __train_init(self, params, train_path, valid_path=None, test_path=None):
  """Create G2P model and initialize or load parameters in session."""

  # Preliminary actions before model creation.
  # Load model parameters.

  if self.model_dir:
    data_utils.save_params(params.n
                           param
                           self.m

  #  準備 G2P 的訓練資料
  print("Preparing G2P data")
 #來自 data_utils 已經準備好的資料
  train_gr_ids, train_ph_ids, valid_gr_i
  self.ph_vocab, self.test_lines =\
  data_utils.prepare_g2p_data(self.m
                              tes
  # Read data into buckets and compute their sizes.
  print ("Reading development and training data.")
```

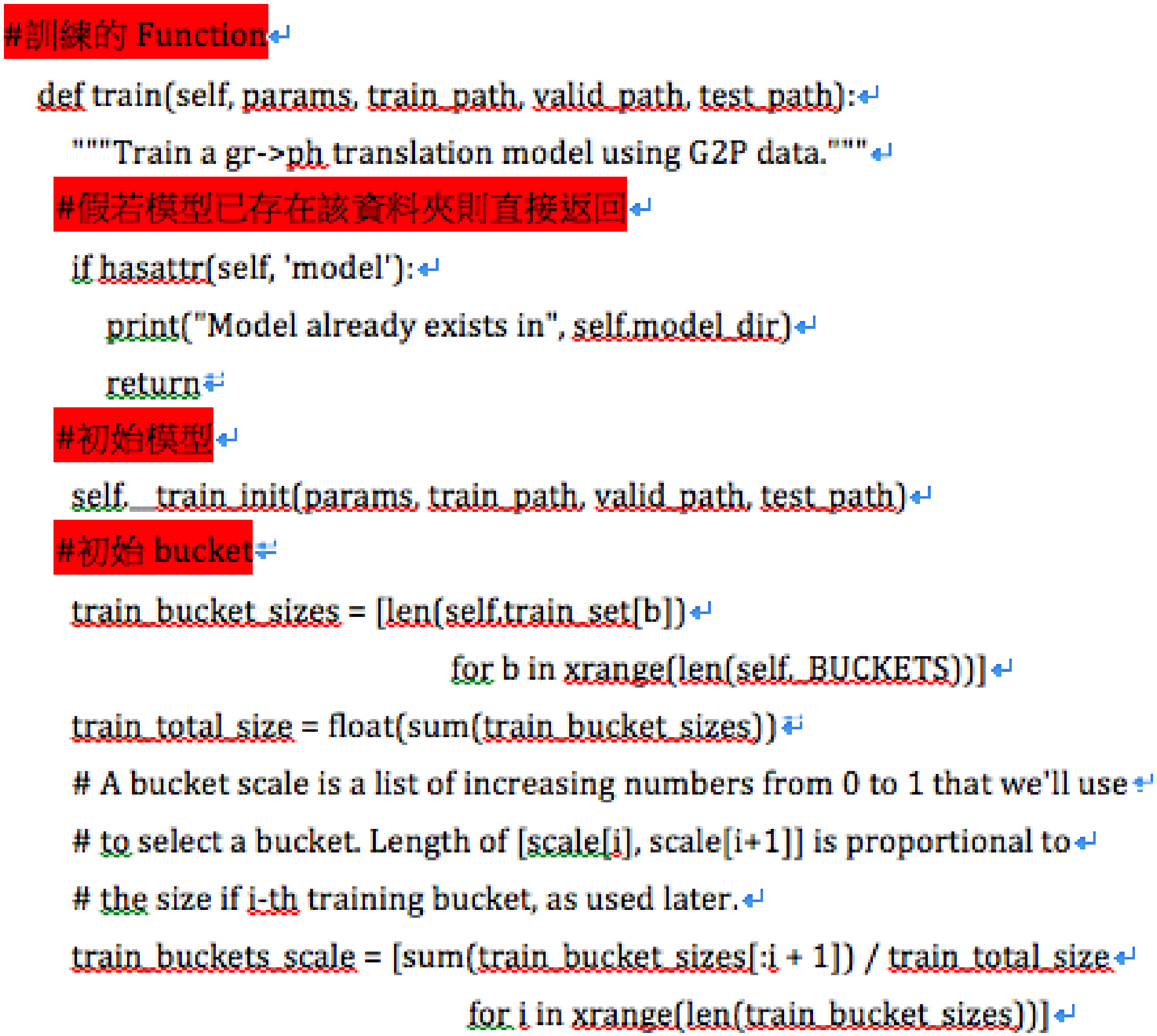
```
  #  生成模型
  print("Creating %d layers of %d units." % (params.num_layers,
params.size))
  self.model = seq2seq_model.Seq2SeqModel(len(self.gr_vocab),
                                          len(self.ph_vocab),
self._BUCKETS, params.size, params.num_layers, params.max_gradient_norm,
                                          params.batch_size,
                                          params.learning_rate,
                                          params.lr_decay_factor,
                                          forward_only=False)
  self.model.saver = tf.train.Saver(tf.all_variables(), max_to_keep=1)
  print("Created model with fresh parameters.")
  self.session.run(tf.initialize_all_variables())
```

Coding Comments-bucket function



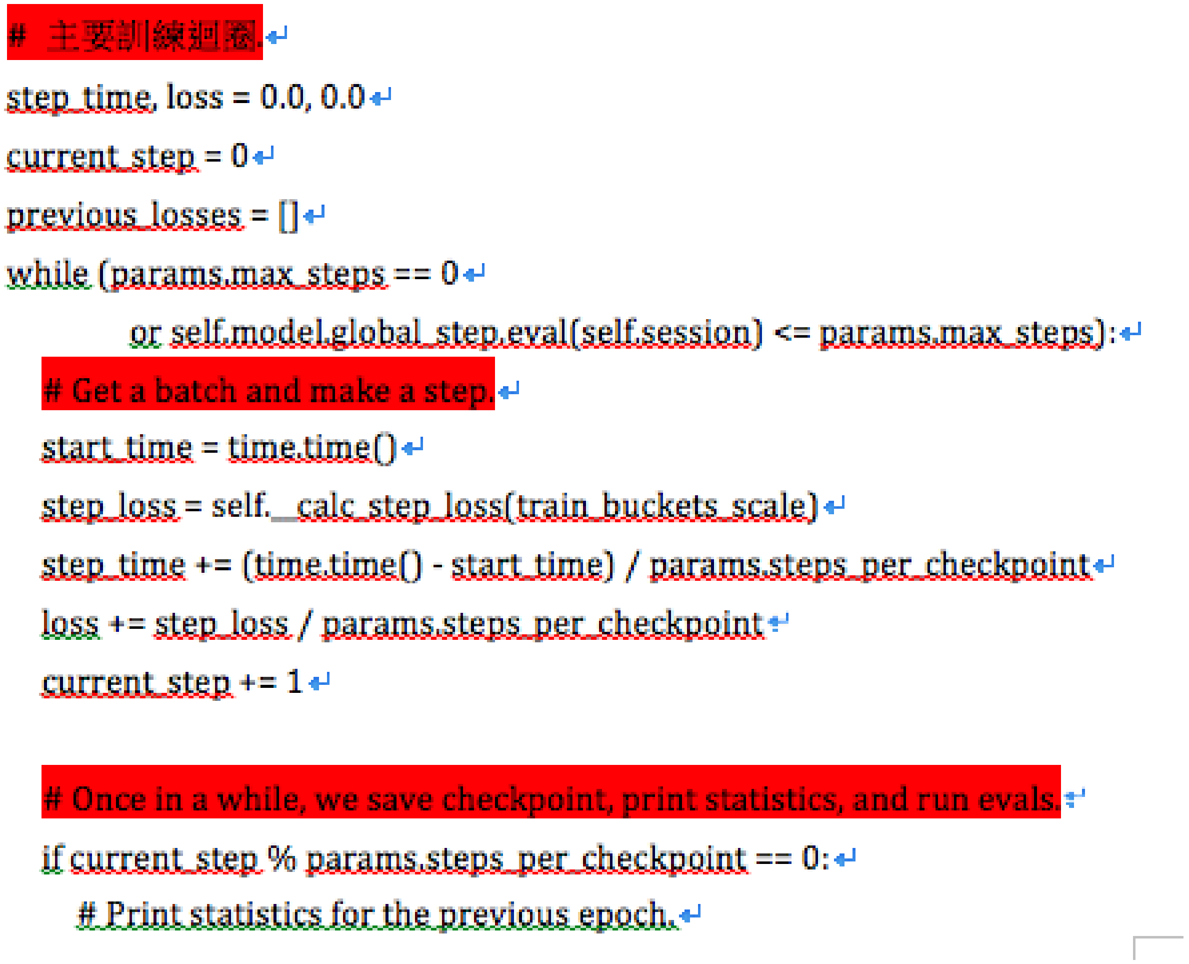
```
#將訓練資料編碼成 bucket 裡的狀態，也就是如上所提補成 bucket 的 size
  def _put_into_buckets(self, source, target):
    # By default unk to unk
    data_set = [[[[4], [4]]] for _ in self._BUCKETS]

    for source_ids, target_ids in zip(source, target):
      target_ids.append(data_utils.EOS_ID)
      for bucket_id, (source_size, target_size) in enumerate(self._BUCKETS):
        if len(source_ids) < source_size and len(target_ids) < target_size:
          data_set[bucket_id].append([source_ids, target_ids])
          break
    return data_set
```

Coding Comments-train function



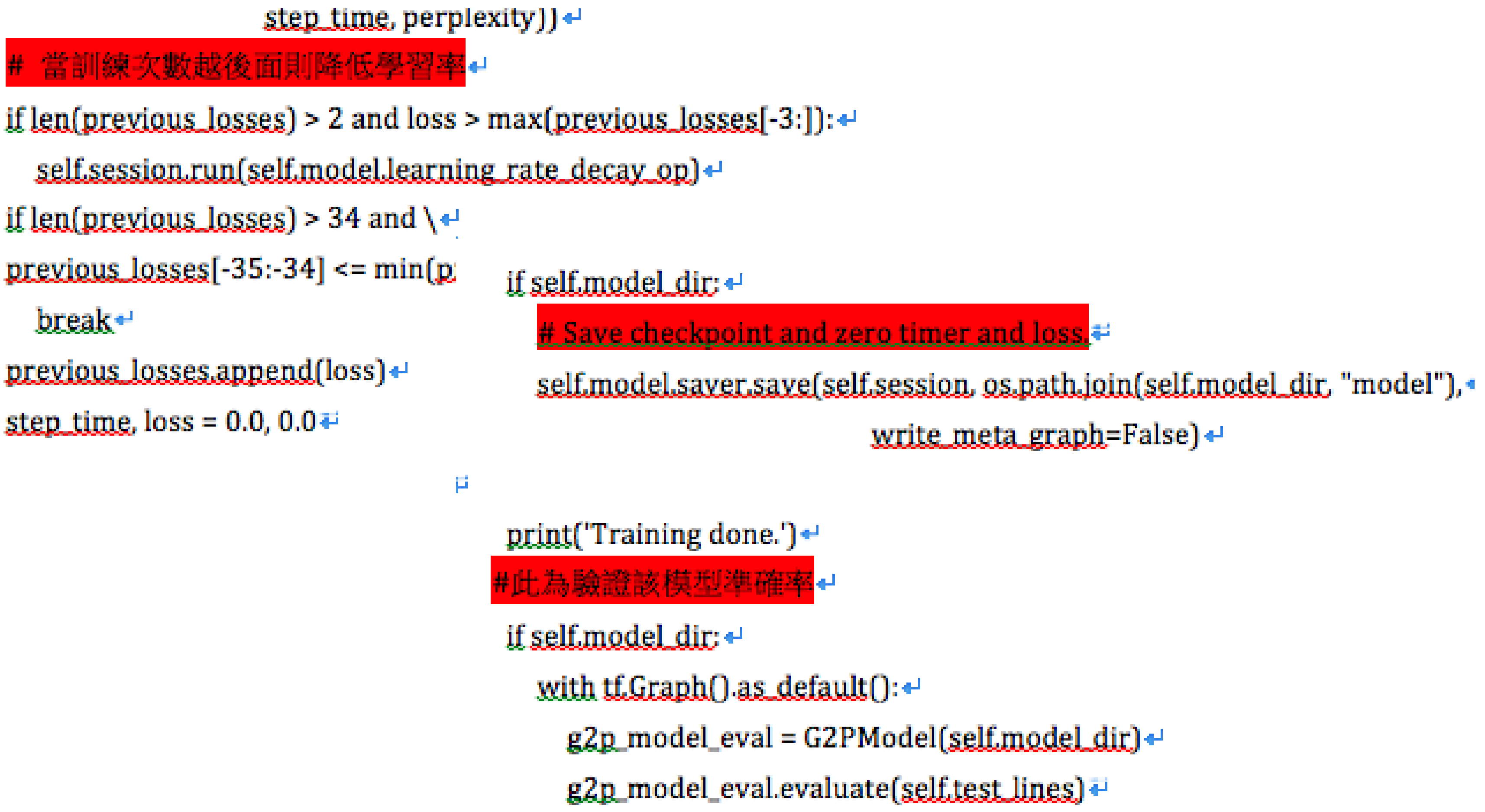
```
#訓練的 Function
  def train(self, params, train_path, valid_path, test_path):
    """Train a gr->ph translation model using G2P data."""
    #假若模型已存在該資料夾則直接返回
    if hasattr(self, 'model'):
      print("Model already exists in", self.model_dir)
      return
    #初始模型
    self.__train_init(params, train_path, valid_path, test_path)
    #初始 bucket
    train_bucket_sizes = [len(self.train_set[b])
                          for b in xrange(len(self._BUCKETS))]
    train_total_size = float(sum(train_bucket_sizes))
    # A bucket scale is a list of increasing numbers from 0 to 1 that we'll use
    # to select a bucket. Length of [scale[i], scale[i+1]] is proportional to
    # the size if i-th training bucket, as used later.
    train_buckets_scale = [sum(train_bucket_sizes[:i + 1]) / train_total_size
                           for i in xrange(len(train_bucket_sizes))]
```

Coding Comments-train function



```
# 主要訓練迴圈.
step_time, loss = 0.0, 0.0
current_step = 0
previous_losses = []
while (params.max_steps == 0
          or self.model.global_step.eval(self.session) <= params.max_steps):
    # Get a batch and make a step.
    start_time = time.time()
    step_loss = self.__calc_step_loss(train_buckets_scale)
    step_time += (time.time() - start_time) / params.steps_per_checkpoint
    loss += step_loss / params.steps_per_checkpoint
    current_step += 1

    # Once in a while, we save checkpoint, print statistics, and run evals.
    if current_step % params.steps_per_checkpoint == 0:
        # Print statistics for the previous epoch.
```

Coding Comments-train function

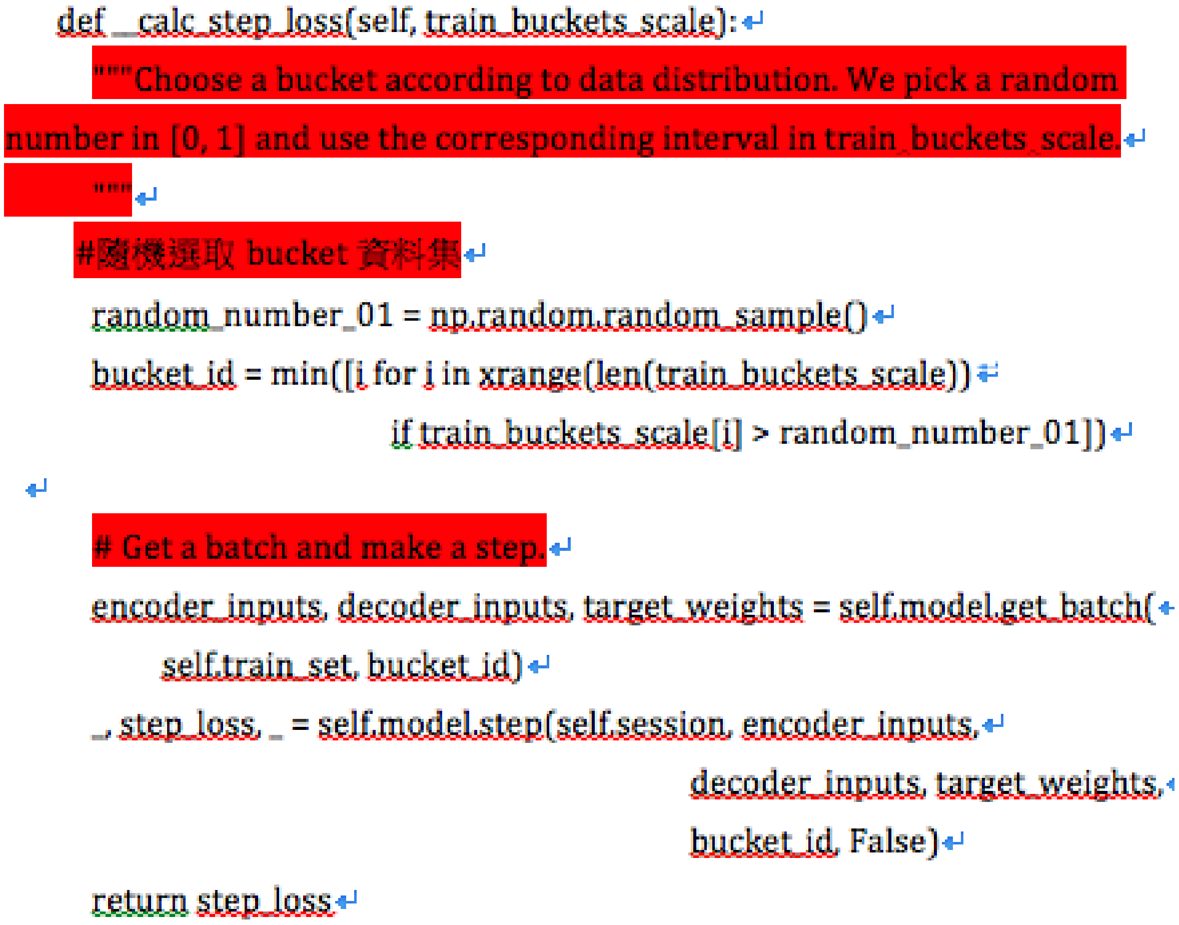


```
                    step_time, perplexity))
#  當訓練次數越後面則降低學習率
if len(previous_losses) > 2 and loss > max(previous_losses[-3:]):
    self.session.run(self.model.learning_rate_decay_op)
if len(previous_losses) > 34 and \
previous_losses[-35:-34] <= min(p
    break
previous_losses.append(loss)
step_time, loss = 0.0, 0.0
```

```
if self.model_dir:
    # Save checkpoint and zero timer and loss.
    self.model.saver.save(self.session, os.path.join(self.model_dir, "model"),
                                  write_meta_graph=False)
```

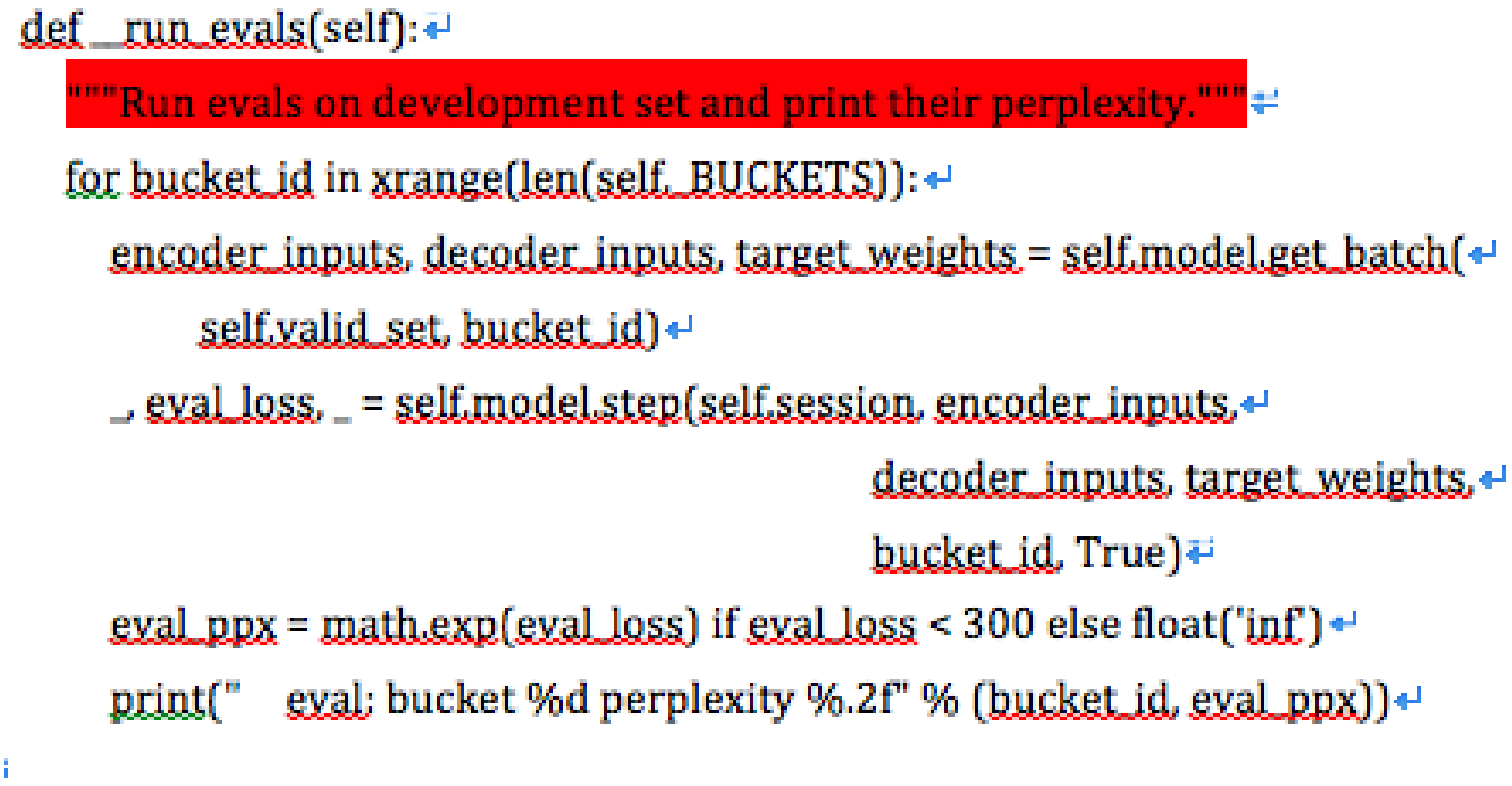
```
print('Training done.')
#此為驗證該模型準確率
if self.model_dir:
    with tf.Graph().as_default():
        g2p_model_eval = G2PModel(self.model_dir)
        g2p_model_eval.evaluate(self.test_lines)
```

Coding Comments-calculate loss function



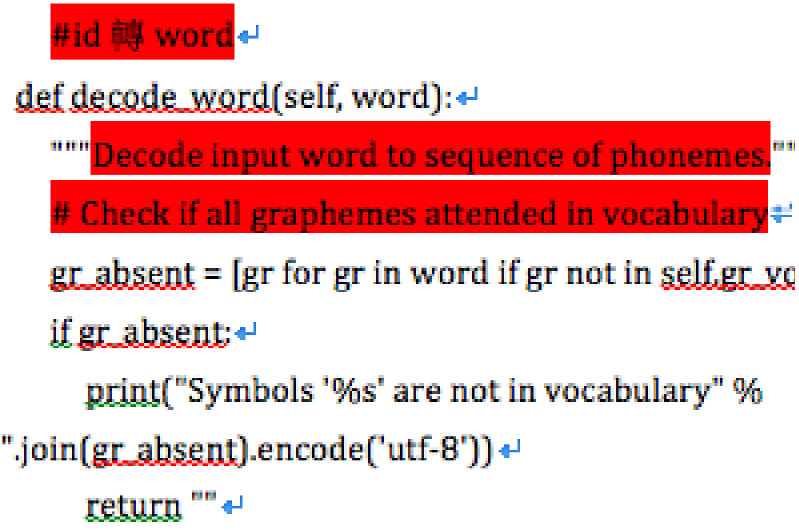
```
def _calc_step_loss(self, train_buckets_scale):
    """Choose a bucket according to data distribution. We pick a random
number in [0, 1] and use the corresponding interval in train_buckets_scale.
    """
   #隨機選取 bucket 資料集
    random_number_01 = np.random.random_sample()
    bucket_id = min([i for i in xrange(len(train_buckets_scale))
                     if train_buckets_scale[i] > random_number_01])

    # Get a batch and make a step.
    encoder_inputs, decoder_inputs, target_weights = self.model.get_batch(
        self.train_set, bucket_id)
    _, step_loss, _ = self.model.step(self.session, encoder_inputs,
                                      decoder_inputs, target_weights,
                                      bucket_id, False)
    return step_loss
```

Coding Comments-evaluate function

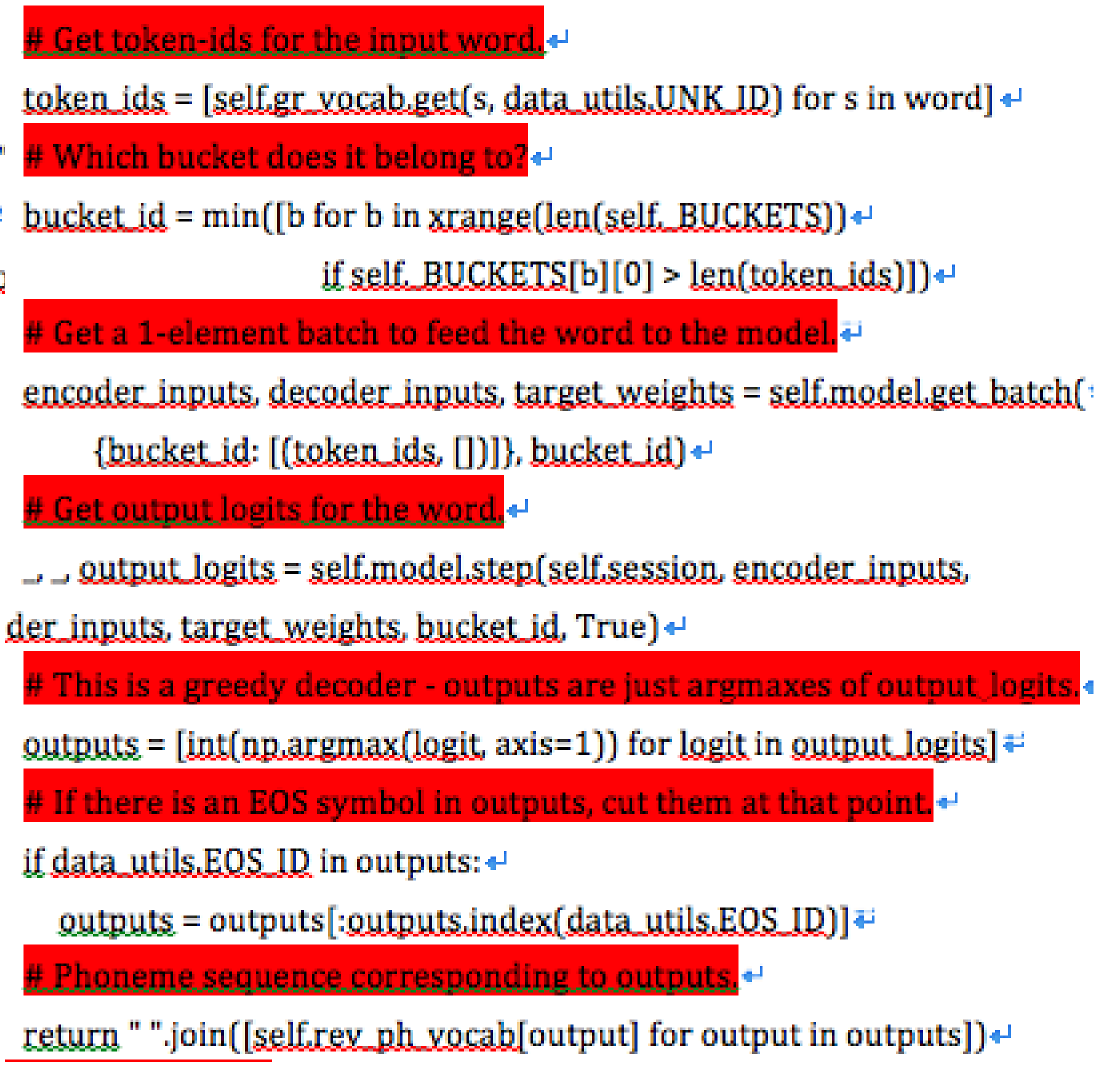


```
def __run_evals(self):
    """Run evals on development set and print their perplexity."""
    for bucket_id in xrange(len(self._BUCKETS)):
        encoder_inputs, decoder_inputs, target_weights = self.model.get_batch(
                self.valid_set, bucket_id)
        _, eval_loss, _ = self.model.step(self.session, encoder_inputs,
                                          decoder_inputs, target_weights,
                                          bucket_id, True)
        eval_ppx = math.exp(eval_loss) if eval_loss < 300 else float('inf')
        print("    eval: bucket %d perplexity %.2f" % (bucket_id, eval_ppx))
```

Coding Comments-ID to Word function

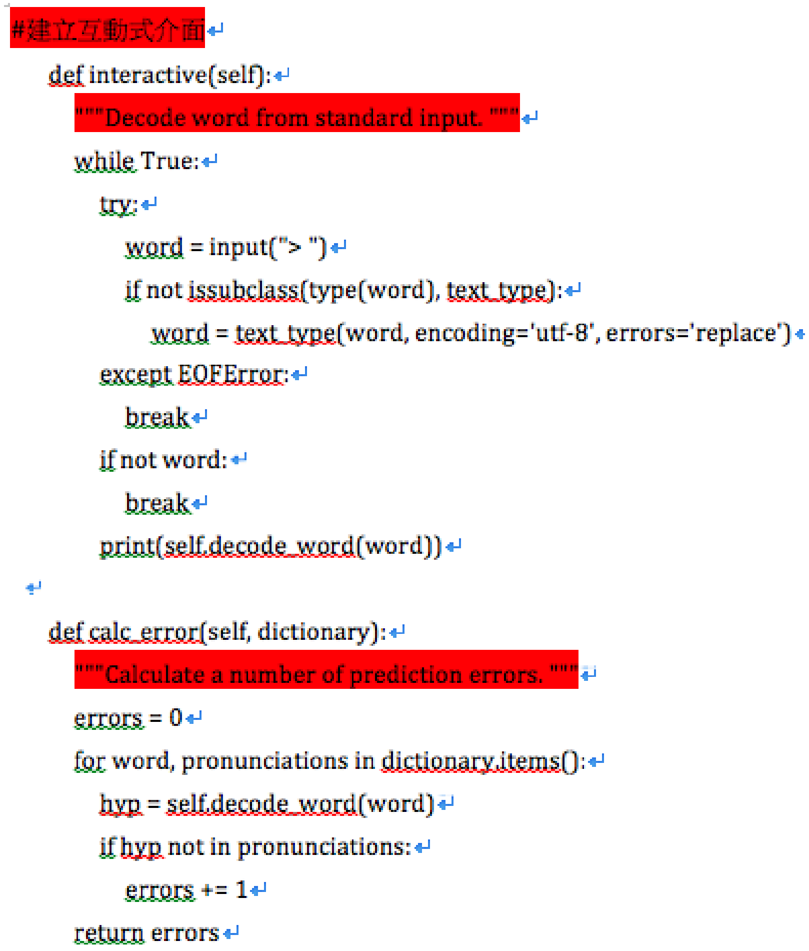


```
#id 轉 word
def decode_word(self, word):
  """Decode input word to sequence of phonemes.""
  # Check if all graphemes attended in vocabulary
  gr_absent = [gr for gr in word if gr not in self.gr_vo
  if gr_absent:
    print("Symbols '%s' are not in vocabulary" %
".join(gr_absent).encode('utf-8'))
    return ""
```



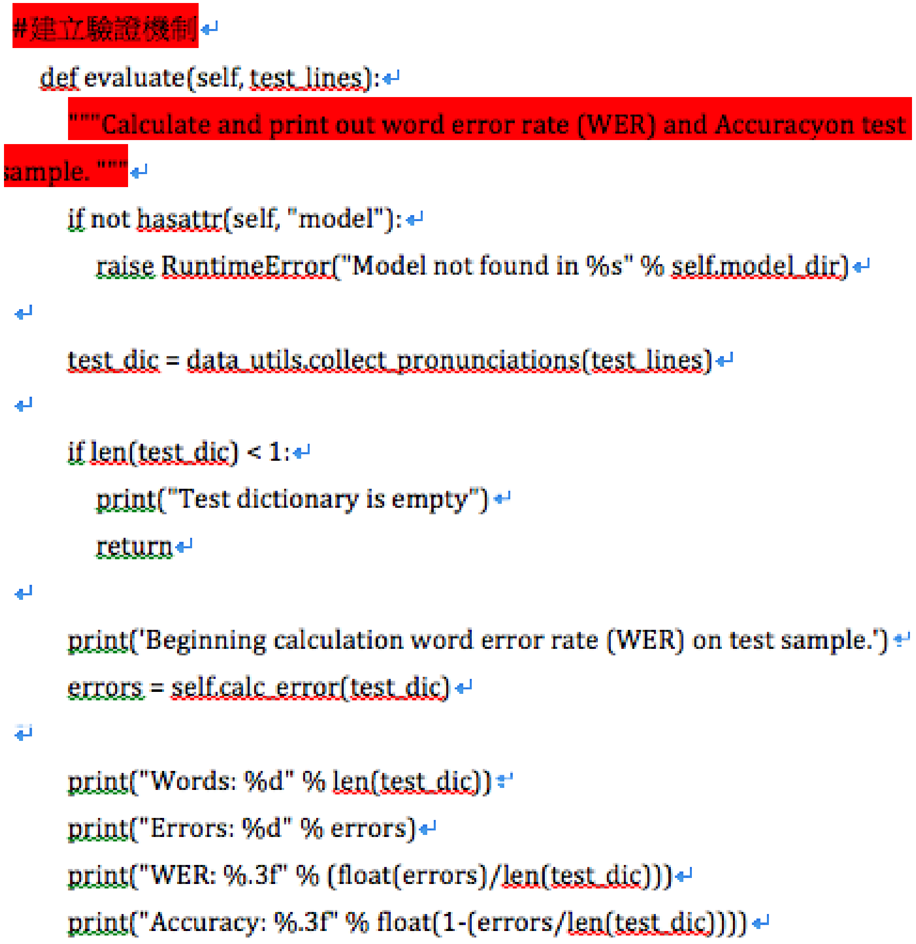
```
# Get token-ids for the input word.
token_ids = [self.gr_vocab.get(s, data_utils.UNK_ID) for s in word]
# Which bucket does it belong to?
bucket_id = min([b for b in xrange(len(self._BUCKETS))
                 if self._BUCKETS[b][0] > len(token_ids)])
# Get a 1-element batch to feed the word to the model.
encoder_inputs, decoder_inputs, target_weights = self.model.get_batch(
    {bucket_id: [(token_ids, [])]}, bucket_id)
# Get output logits for the word.
_, _, output_logits = self.model.step(self.session, encoder_inputs,
der_inputs, target_weights, bucket_id, True)
# This is a greedy decoder - outputs are just argmaxes of output_logits.
outputs = [int(np.argmax(logit, axis=1)) for logit in output_logits]
# If there is an EOS symbol in outputs, cut them at that point.
if data_utils.EOS_ID in outputs:
  outputs = outputs[:outputs.index(data_utils.EOS_ID)]
# Phoneme sequence corresponding to outputs.
return " ".join([self.rev_ph_vocab[output] for output in outputs])
```

Coding Comments-Interactive&calculate error function



```
#建立互動式介面
    def interactive(self):
        """Decode word from standard input. """
        while True:
            try:
                word = input("> ")
                if not issubclass(type(word), text_type):
                    word = text_type(word, encoding='utf-8', errors='replace')
            except EOFError:
                break
            if not word:
                break
            print(self.decode_word(word))

    def calc_error(self, dictionary):
        """Calculate a number of prediction errors. """
        errors = 0
        for word, pronunciations in dictionary.items():
            hyp = self.decode_word(word)
            if hyp not in pronunciations:
                errors += 1
        return errors
```

Coding Comments-evaluate function

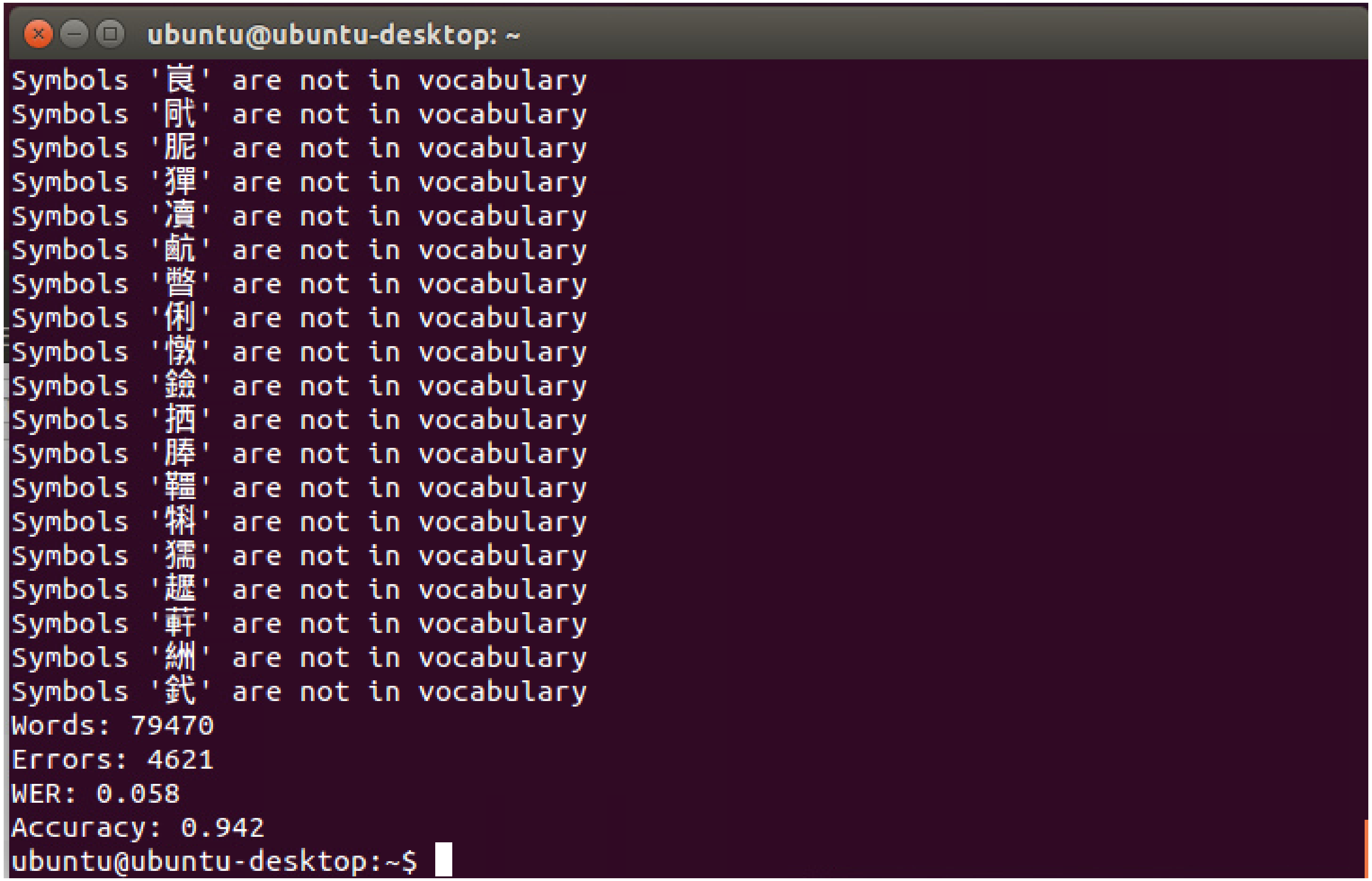


```
#建立驗證機制
    def evaluate(self, test_lines):
        """Calculate and print out word error rate (WER) and Accuracyon test
sample. """
        if not hasattr(self, "model"):
            raise RuntimeError("Model not found in %s" % self.model_dir)

        test_dic = data_utils.collect_pronunciations(test_lines)

        if len(test_dic) < 1:
            print("Test dictionary is empty")
            return

        print('Beginning calculation word error rate (WER) on test sample.')
        errors = self.calc_error(test_dic)

        print("Words: %d" % len(test_dic))
        print("Errors: %d" % errors)
        print("WER: %.3f" % (float(errors)/len(test_dic)))
        print("Accuracy: %.3f" % float(1-(errors/len(test_dic))))
```

Model

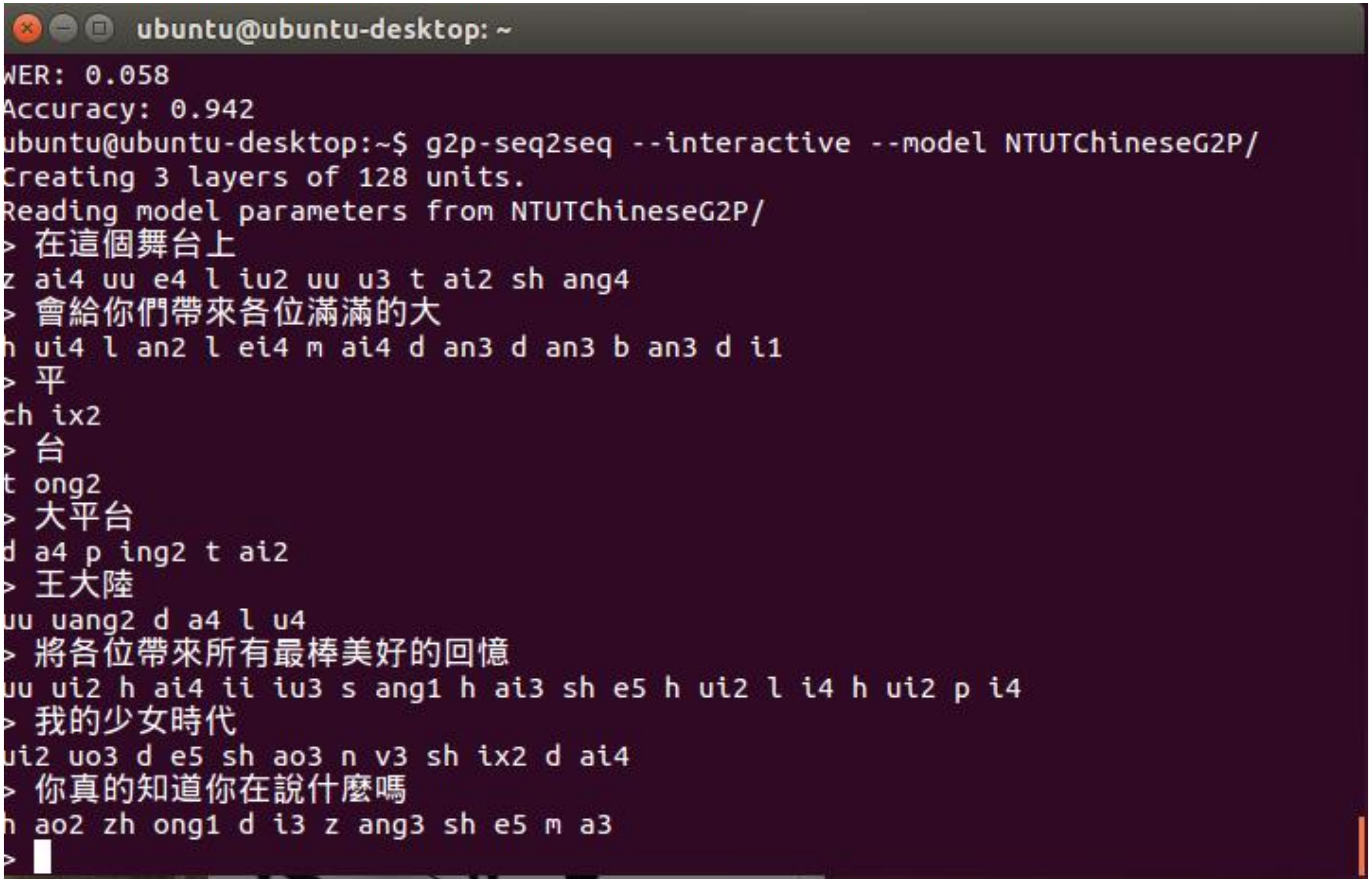
- The model have 3 layer with 128 units.
- It trained about 6 hours.

Result-evaluate



```
Symbols '崀' are not in vocabulary
Symbols '戙' are not in vocabulary
Symbols '胒' are not in vocabulary
Symbols '𤢂' are not in vocabulary
Symbols '瀆' are not in vocabulary
Symbols '魧' are not in vocabulary
Symbols '瞥' are not in vocabulary
Symbols '俐' are not in vocabulary
Symbols '憿' are not in vocabulary
Symbols '鐱' are not in vocabulary
Symbols '𢪃' are not in vocabulary
Symbols '胮' are not in vocabulary
Symbols '輻' are not in vocabulary
Symbols '料' are not in vocabulary
Symbols '獳' are not in vocabulary
Symbols '趯' are not in vocabulary
Symbols '蓒' are not in vocabulary
Symbols '絒' are not in vocabulary
Symbols '鈛' are not in vocabulary
Words: 79470
Errors: 4621
WER: 0.058
Accuracy: 0.942
ubuntu@ubuntu-desktop:~$ 
```

Result-interactive



```
WER: 0.058
Accuracy: 0.942
ubuntu@ubuntu-desktop:~$ g2p-seq2seq --interactive --model NTUTChineseG2P/
Creating 3 layers of 128 units.
Reading model parameters from NTUTChineseG2P/
> 在這個舞台上
z ai4 uu e4 l iu2 uu u3 t ai2 sh ang4
> 會給你們帶來各位滿滿的大
h ui4 l an2 l ei4 m ai4 d an3 d an3 b an3 d i1
> 平
ch ix2
> 台
t ong2
> 大平台
d a4 p ing2 t ai2
> 王大陸
uu uang2 d a4 l u4
> 將各位帶來所有最棒美好的回憶
uu ui2 h ai4 ii iu3 s ang1 h ai3 sh e5 h ui2 l i4 h ui2 p i4
> 我的少女時代
ui2 uo3 d e5 sh ao3 n v3 sh ix2 d ai4
> 你真的知道你在說什麼嗎
h ao2 zh ong1 d i3 z ang3 sh e5 m a3
> 
```

Thank you!