logits: 非空的tensor，类型必须为half, float32, float64.

**Sampled softmax and output projection**

This is a faster way to train a softmax classifier over a huge number of classes.This operation is for training only. It is generally an underestimate of the full softmax loss.At inference time, you can compute full softmax probabilities with the expression tf.nn.softmax(tf.matmul(inputs, tf.transpose(weights)) + biases).

softmax计算一个词的概率的时候需要归一化这样就要花费大量的计算量，为了解决这一问题，所以采用sample softmax，它是使用「importance sampling」+「sampled softmax」，当然采用sample softmax的时候需要先比较 number of samples和target vocabulary size的大小，只有number of samples远小于target vocabulary size时才需采用sample softmax，如果target vocabulary size很小（默认小于512），那么最好采用standard softmax loss.但是采用sampled softmax的时候输出维度会下降預設為512維，所以为了译码的时候回归到vocabulary size，我们需要进行输出结果的映射output projection，透過一個output\_projection(一組W及b)，將較小維度的output(預設為512維)，投影成原先的40,000維的大小。。

w = tf.get\_variable("proj\_w", [size, self.target\_vocab\_size])  
 w\_t = tf.transpose(w)  
 b = tf.get\_variable("proj\_b", [self.target\_vocab\_size])

output\_projection = (w, b)

if output\_projection is not None:  
  self.outputs[b] = [tf.matmul(output, output\_projection[0]) +  
                     output\_projection[1] for ...]

Bucketing and padding

雖然理論上RNN可以處理任意長度的句子，但實際上卻會產出相當多個graph，以配合各種英文句子長度vs.法文句子長度的組合。因此，本模型預先設定數個句長組合的bucket：

|  |  |
| --- | --- |
| 1 | buckets = [(5, 10), (10, 15), (20, 25), (40, 50)] |

使模型僅需產出上述幾個組合的graph即可，而不必產出大量且可能有不少重複的graph。

也就是說，若有一英法句子對的長度為(8, 18)，就會被分到「大於該長度對」且「bucket中的長度對最小」的一組，也就是(20, 25)。而句子不夠填滿的部分就用<PAD>填滿為止，例如編碼後長度為3的一句句子：[2, 17, 4]，就會被 padding 到長度為5為止：[2, 17, 4, PAD, PAD]