主要函数结构

先来看看主要函数的调用过程（接下来也是按照这样的结构解释源码）：

1. embedding_attention_seq2seq()
- 2. embedding_attention_decoder()
- 3. attention_decoder()
- attention()

其中函数 attention()是实现每一个时刻的 attention 机制的主要函数。

1. embedding_attention_seq2seq()

embedding_attention_seq2seq 函数参数说明如下：

```
def embedding_attention_seq2seq(encoder_inputs,# 编码器的输入
                                decoder_inputs,# 解码器的输入
                                cell,# 所用的 cell
                                num_encoder_symbols,# 编码的符号总数
                                num_decoder_symbols,# 解码的符号总数
                                embedding_size,# 向量的维度
                                num_heads=1,# 这个也不知道啊，默认为 1，基本上可以当它不存在了
```

output_projection=None, # 输出投影

feed_previous=False,# 当前输入是否要考虑前一个时刻的输出，这个一般在训练的时候为 False，预测的时候为 True

dtype=None,

scope=None,

initial_state_attention=False)

返回值：

由（outputs，state）组成的元组

outputs 的 shape 为[batch_size x num_decoder_symbols]

state 为最后一个时刻 decoder 的状态，shape 为[batch_size x cell.state_size]

这个函数首先为一个编码的过程：

使用 EmbeddingWrapper 将输入映射到 embedding_size 大小的向量，然后通过调用 static_rnn 得到了 encoder 的每一个时刻的输出，即为之后我们需要 attention 的向量。

然后再进行解码的过程：

首先使用 OutputProjectionWrapper 将解码器的输出映射成想要的维度

接下来执行：

```
if isinstance(feed_previous, bool):
return embedding_attention_decoder
```