**实验十 神经机器翻译**

1. 实验目的
2. 掌握循环神经网络及其变种的结构。
3. 了解Encoder-Decoder框架， word2vec技术，以及自然语言处理中的attention机制。
4. 进一步熟悉TensorFlow的使用，学会使用其自带的RNN框架、或LSTM框架，并结合收集到的相关数据，搭建Encoder-Decoder机器翻译模型。
5. 实验要求

利用Python语言和TensorFlow框架，构造简单的机器翻译模型，以实现英语和汉语的相互转换。

1. 实验原理

循环神经网络相关概念和原理参考《深度学习》课程讲授内容。

Encoder-Decoder模型是使用神经网络进行机器翻译的基本方法，一般也称作seq2seq模型。具体来说，它会先用一个Encoder将输入的序列编码为一个上下文向量c，再使用Decoder对c进行解码（将c作为初始状态的输入，或将c当作每一步的输入），将之变为输出序列，如图1、图2所示。在机器翻译任务中，输入序列为待翻译的英文，输出序列为其对应的汉语，Encoder和Decoder分别为两个RNN。



图1 c作为初始状态



图2 c作为每一步的输入

在传统的Encoder-Decoder结构中，c中必须包含原始序列中的全部信息，它的长度是限制模型性能的主要瓶颈。注意力机制（Attention）通过在每个时间输入不同的c来解决这个问题，每一个c会自动选取与当前所要输出的y最合适的上下文信息，即，用来衡量Encoder中第j阶段的和Decoder中第i阶段的相关性。是模型在训练时自动学出来的，且满足。如图3所示。



图3 带有注意力的Decoder

1. 实验所用工具及数据集

本实验主要针对中英机器翻译，使用的数据库来自NiuTrans提供的开源中英平行语料库，包含中、英文各10万条。如图4所示。

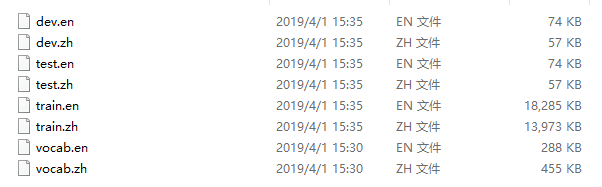


图4 NiuTrans语料库

数据集包含四部分：训练集（train.zh，train.en）、验证集（dev.zh，dev.en）、测试集（test.zh，test.en），以及词汇表（vocab.zh，vocab.en），其中以.en为后缀的文件为英语语料，.zh为后缀的为中文语料。值得注意的是，每个词汇表都会包含三个特殊的单词：<unk>、<s>、</s>。<unk>表示所有不常见的单词（对文本中使用的单词进行计数，取最常用的单词训练，剩下的单词都被替换为<unk>），<s>表示句子的开头，</s>表示句子的结尾。

本实验用到的数据集已经做好了中文分词，中文的数据样例如下：

北约 不少 飞机 不得不 携 弹 返航 , 降低 了 军事 能力 的 使用 效能 , 增加 了 战斗 成本 .

每个词之间用空格分隔，标点符号也算作一个单词。相应的英文样例如下：

many nato planes had to return to base laden with munitions , thus lowering the efficiency of use of military power and increasing the costs of fighting .

由于英语中每个单词之间都有空格，故不需要分词，需要做的就是把首句大写字母还原为小写。

1. 实验步骤与方法

示例模型结构可以分为4个部分。首先是将输入序列进行词嵌入，再将词嵌入后的向量输入到LSTM中编码以聚合输入序列的上下文信息。得到的上下文向量输入到解码器中进行解码，解码器也是由LSTM组成，最终输出翻译结果，如图5所示。下面针对这几层对部分代码片段进行介绍。



图5 模型结构图

（1）词嵌入

embdding\_ops.embedding\_lookup是tensorflow中的方法，用于查找某个单词id对应的词嵌入向量。这里的encoder\_emb\_inp是词嵌入之后得到的向量，将被传入到encoder层中去。

encoder\_emb\_inp = tf.nn.embedding\_lookup(self.embedding\_encoder, source)

（2）encoder层

Encoder层可以使用单向或双向的RNN或者LSTM。以单向的RNN为例：

cell = self.\_build\_encoder\_cell(

hparams, num\_layers, num\_residual\_layers)

encoder\_outputs, encoder\_state = tf.nn.dynamic\_rnn(

cell, encoder\_emb\_inp, dtype=dtype,

sequence\_length=iterator.source\_sequence\_length,

time\_major=self.time\_major)

（3）decoder层

#Bulild RNN cell

decoder\_cell = tf.nn.rnn\_cell.BasicLSTMCell(num\_units)

#Helper

helper = tf.contrib.seq2seq.TrainingHelper(

decoder\_emb\_inp, decoder\_lengths, time\_major=True)

#Decoder

decoder = tf.contrib.seq2seq.BasicDecoder(

decoder\_cell, helper, encoder\_state,

output\_layer=projection\_layer)

#Dynamic decoding

outputs, \_ = tf.contrib.seq2seq.dynamic\_decode(decoder, ...)

logits = outputs.rnn\_output

这里的核心是tf.contrib.seq2seq.BasicDecoder，它是Tensorflow seq2seq API中的一个类，它接收一个decoder\_cell，一个helper，还有上下文变量encode\_state,投影层变量projection\_layer，最终利用它得到结果。投影层变量的定义如下，它把LSTM中的每一步输出转换为一个vocab\_size维的向量，vocab\_size是目标语言词汇表中所有词的个数。

projection\_layer = layers\_core.Dense( tgt\_vocab\_size, use\_bias=False)

（4）注意力

除了基本的Encoder-Decoder模型外，还可以在模型中添加注意力机制以提升模型表现。

#attention\_states: [batch\_szie, max\_time, num\_units]

attention\_states = tf.transpose(encoder\_outputs, [1, 0, 2])

#Creat an attention mechansim

attention\_mechanism = tf.contrib.seq2seq.LuongAttention(

num\_units, attention\_states,

memory\_sequence\_length=source\_sequence\_length)

decoder\_cell = tf.contrib.seq2seq.AttentionWrapper(

decoder\_cell, attention\_mechanism,

attention\_layer\_size=num\_units)