

网络空间先进技术研究院

**机器学习与人工智能**

**课程报告**

项目名称 RNN神经机器翻译

学 院 计算机科学与教育软件学院

专业班级 计算机技术

学 号 2111806028

姓 名 胡卫雄

任课教师 仇晶

2018 年 11 月

**课程报告评定表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学生姓名 | 胡卫雄 | 学号 | 2111806028 | 成绩 |  |
| 专业班级 | 计算机技术 | | | | |
| 项目名称 | RNN神经机器翻译 | | | | |
| 指  导  教  师  评  语 | 指导教师：  年 月 日 | | | | |

目录

[1项目目的 4](#_Toc533357634)

[2项目环境和条件 4](#_Toc533357635)

[3项目原理 4](#_Toc533357636)

[4项目内容 5](#_Toc533357637)

[4.1 Encoder-Decoder模型的深度理解 5](#_Toc533357638)

[4.2 Encoder-Decoder模型在机器翻译任务的应用 5](#_Toc533357639)

[4.3 Keras深度学习库的使用及理解 5](#_Toc533357640)

[4.4 LSTM的使用 5](#_Toc533357641)

[5项目过程与内容 6](#_Toc533357642)

[5.1任务分析 6](#_Toc533357643)

[5.2数据分析 6](#_Toc533357644)

[5.3开发步骤 7](#_Toc533357645)

[5.3.1 数据预处理 7](#_Toc533357646)

[5.3.2 encoder-decoder模型的搭建 8](#_Toc533357647)

[5.3.3 encoder-decoder模型的搭建 10](#_Toc533357648)

[5.4关键问题 11](#_Toc533357649)

[5.5 实验结果分析 11](#_Toc533357650)

[参考文献 13](#_Toc533357651)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1项目目的 熟悉RNN算法及其在机器翻译任务的应用，理解BPTT算法，encoder-decoder结构，LSTM，seq2seq模型（属于encoder-decoder结构的一种）的相关知识，了解和使用Keras平台/开发环境（如TensorFlow/Keras等）。 2项目环境和条件 软硬件环境和条件：PC 机一台，Window 7 操作系统，Python 3.5, Keras 深度学习库，pandas , numpy，中英文训练语料集。 3项目原理 RNN算法原理：RNN是Hopfield在1982年提出的一种能够处理时序问题的循环神经网络[1]。RNNs的目的使用来处理序列数据。在传统的神经网络模型中，是从输入层到隐含层再到输出层，层与层之间是全连接的，每层之间的节点是无连接的。但是这种普通的神经网络对于很多问题却无能无力。例如，你要预测句子的下一个单词是什么，一般需要用到前面的单词，因为一个句子中前后单词并不是独立的。RNNs之所以称为循环神经网路，即一个序列当前的输出与前面的输出也有关。具体的表现形式为网络会对前面的信息进行记忆并应用于当前输出的计算中，即隐藏层之间的节点不再无连接而是有连接的，并且隐藏层的输入不仅包括输入层的输出还包括上一时刻隐藏层的输出。理论上，RNNs能够对任何长度的序列数据进行处理。但是在实践中，为了降低复杂性往往假设当前的状态只与前面的几个状态相关。  LSTM算法原理：LSTM（Long Short-Term Memory）是长短期记忆网络，是一种时间递归神经网络，适合于处理和预测[时间序列](https://baike.baidu.com/item/%E6%97%B6%E9%97%B4%E5%BA%8F%E5%88%97/1389644" \t "https://baike.baidu.com/item/LSTM/_blank)中间隔和延迟相对较长的重要事件。采用了特殊隐式单元的LSTM（long short-term memory networks）其自然行为便是长期的保存输入。一种称作记忆细胞的特殊单元类似累加器和门控神经元：它在下一个时间步长将拥有一个权值并联接到自身，拷贝自身状态的真实值和累积的外部信号，但这种自联接是由另一个单元学习并决定何时清除记忆内容的乘法门控制的。LSTM网络随后被证明比传统的RNNs更加有效，尤其当每一个时间步长内有若干层时，整个语音识别系统能够完全一致的将声学转录为字符序列。目前LSTM网络或者相关的门控单元同样用于编码和解码网络，并且在机器翻译中表现良好。  Seq2Seq原理：Seq2Seq是一个Encoder-Decoder结构的模型，输入是一个序列，输出也是一个序列。Encoder将一个可变长度的输入序列变为固定长度的向量，Decoder将这个固定长度的向量解码成可变长度的输出序列。 4项目内容4.1 Encoder-Decoder模型的深度理解 Encoder-Decoder的模型虽然非常经典，但是也存在很大的局限性。最大的局限性在于编解码之间唯一的联系就是一个固定长度的语义向量C。也就是说，编码器要将整个输入序列的信息压缩进一个固定的向量中去。但是这样做有两个弊端，一是语义向量无法完全表示整个序列的信息，还有就是先输入的内容携带的信息会被后输入的信息稀释掉，或者说覆盖掉。输入序列越长，这个现象就越严重。这就使得在解码的时候一开始就没有获得输入序列足够的信息，那么解码的准确度就会大打折扣。所以后面有了Attention机制的产生，Attention机制通过给“合适”的部位分配更多的注意力（权重）来更好地获取输入信息。 4.2 Encoder-Decoder模型在机器翻译任务的应用 Encoder-Decoder模型使用两个RNN网络来完成机器翻译工作，第一个RNN网络把一串符号序列编码成一个固定长度的语义向量来表示，第二个RNN网络把这个固定长度的语义向量解码成目标符号序列。通过联合训练这两个RNN网络，使得对于输入序列，得到输出序列条件概率的最大化。我们在此次机器翻译的任务中RNN网络使用的是LSTM。LSTM能学到序列中距离更远的两个元素之间的关联信息，而且层数越多，能记忆的信息也越多，所以处理长句子是效果优越。 4.3 Keras深度学习库的使用及理解 Keras是基于Theano,Tensorflow,CNTK的深度学习库，具有简易和快速的原型设计（keras具有高度模型化，极简和可扩充性等特点）；支持CNN和RNN，或者二者的结合；执行任意的链接方案（包括多输入和多输出训练）；支持无缝CPU和GPU切换。 4.4 LSTM的使用 LSTM有三个 gate，第一个 input gate，当 current cell 很重要的时候，就希望更新它，i\_t 的值就很大。h\_(t-1)就代表着 memory。第二个 forget gate，就是忘记你的过去吧，忘记你至今学到的所有。第三个 output gate，有了它，你就不需要把内部的状态告诉后面的网络，只需要keep around就可以了，不需要tell。当你想要预测单词时，当你看到一整句话时，你只需要输出一个结果，并不需要释放所有的隐藏层和记忆。给了两个facts，然后进行推理，再给出output即可。本次实验主要是调用了keras的layers中的LSTM，编码器和解码器都是使用了LSTM来完成的。 5项目过程与内容5.1任务分析 本次实验的任务是使用RNN模型来完成中英文机器翻译的工作。掌握使用keras来构建RNN模型的过程，掌握数据预处理的方法，理解RNN实现机器翻译的原理。 5.2数据分析 表5-1数据集格式   |  | | --- | | When I was in London last year, someone broke into my room and stole my wallet. 當我去年在倫敦的時候，有人闖進了我的房間，偷走了我的錢包。  At the time there were no native English speakers teaching in any public school. 當時沒有任何以英語為母語的人在公立學校任教。  Rio de Janeiro is perfectly safe as long as you stay out of the dangerous areas. 如果你远离危险区域，里约热内卢就是完全安全的。  She was asked to convince him to get his son or someone else to paint the house. 她被要求去说服他以让他或者他的儿子或者是别的人来粉刷屋子。  Even though I studied English for 6 years in school, I'm not good at speaking it. 尽管我在学校学了6年英语，我还是说不好。 |   本次实验使用的数据集是中英文数据集，来实现字符级的seq2seq模型的训练。该文件来自于http://www.manythings.org/anki/,一共包括20133条中文对照翻译的记录。由于硬件条件的限制，本次实验的总共只选取了10000条数据，前8000条记录作为训练集，后2000条作为测试集。数据集的格式上表所示，数据集是每一句英文后对应着一句中文的翻译。后面我们需要将中文和英文切分开，分别加载进相应的列表中以便于后面的利用。 5.3开发步骤5.3.1 数据预处理  |  | | --- | | # 读取文件  data\_path = 'cmn-eng/cmn.txt'  df = pd.read\_table(data\_path,header=None).iloc[:NUM\_SAMPLES,:,]  df.columns=['inputs','targets']  # 将每句的中文句首加上’\t’作为起始标志，句末加上’\n’作为终止标志  df['targets'] = df['targets'].apply(lambda x: '\t'+x+'\n')  input\_texts = df.inputs.values.tolist() # 英文句子列表  target\_texts = df.targets.values.tolist() # 中文句子列表  # 确定中英文各自包含的字符。df.unique()直接取sum可将unique数组中的各个句子拼接成一个长句子  input\_characters = sorted(list(set(df.inputs.unique().sum())))  target\_characters = sorted(list(set(df.targets.unique().sum()))) |   如上所示我们首先是将数据集的前10000条数据用pandas读取出来，然后将数据处理成Keras模型中接受的三维向量。这里需要处理3个向量，分别是encoder的输入encoder\_input,decoder的输入和输出decoder\_input,decoder\_output.   |  | | --- | | INUPT\_LENGTH = max([len(i) for i in input\_texts])  OUTPUT\_LENGTH = max([len(i) for i in target\_texts])  INPUT\_FEATURE\_LENGTH = len(input\_characters)  OUTPUT\_FEATURE\_LENGTH = len(target\_characters)  encoder\_input=np.zeros((NUM\_SAMPLES,INUPT\_LENGTH,INPUT\_FEATURE\_LENGTH),dtype=np.float32)  decoder\_input=np.zeros((NUM\_SAMPLES,OUTPUT\_LENGTH,OUTPUT\_FEATURE\_LENGTH),dtype=np.float32)  decoder\_output=np.zeros((NUM\_SAMPLES,OUTPUT\_LENGTH,OUTPUT\_FEATURE\_LENGTH),dtype=np.float32) |   我们每条句子经过对字母转换成one-hot编码后，生成了LSTM需要的三维输入。其中变量信息如下所示  NUM\_SAMPLES，样本条数，这里是输入的句子条数  INPUT\_LENGTH，输入数据的时刻t的长度，这里为最长的英文句子长度  OUTPUT\_LENGTH，输出数据的时刻t的长度，这里为最长的中文句子长度  INPUT\_FEATURE\_LENGTH，每个时刻进入encoder的lstm单元的数据xt的维度，这里为英文中出现的字符数  OUTPUT\_FEATURE\_LENGTH，每个时刻进入decoder的lstm单元的数据xt的维度，这里为中文中出现的字符数  下面我们将对句子进行字符级one-hot编码，将输入输出数据向量化   |  | | --- | | for seq\_index,seq in enumerate(input\_texts):  for char\_index, char in enumerate(seq):  encoder\_input[seq\_index,char\_index,input\_dict[char]] = 1  for seq\_index,seq in enumerate(target\_texts):  for char\_index,char in enumerate(seq):  decoder\_input[seq\_index,char\_index,target\_dict[char]] = 1.0  if char\_index > 0:  decoder\_output[seq\_index,char\_index-1,target\_dict[char]] = 1.0 |   其中input\_dict和target\_dict为中英文字符与其索引的对应字典；input\_dict\_reverse和target\_dict\_reverse与之相反，索引为键，字符为值   |  | | --- | | input\_dict = {char:index for index,char in enumerate(input\_characters)}  input\_dict\_reverse = {index:char for index,char in enumerate(input\_characters)}  target\_dict = {char:index for index,char in enumerate(target\_characters)}  target\_dict\_reverse = {index:char for index,char in enumerate(target\_characters)} |  5.3.2 encoder-decoder模型的搭建 1）用于训练的模型，可以直接调用keras中的LSTM函数   |  | | --- | | #训练阶段  #encoder  encoder\_input = Input(shape = (None, n\_input))  #encoder输入维度n\_input为每个时间步的输入xt的维度，这里是用来one-hot的英文字符数  encoder = LSTM(n\_units, return\_state=True)  #n\_units为LSTM单元中每个门的神经元的个数，return\_state设为True时才会返回最后时刻的状态h,c  \_,encoder\_h,encoder\_c = encoder(encoder\_input)  encoder\_state = [encoder\_h,encoder\_c]  print(encoder\_state)  #保留下来encoder的末状态作为decoder的初始状态    #decoder  decoder\_input = Input(shape = (None, n\_output))  #decoder的输入维度为中文字符数  decoder = LSTM(n\_units,return\_sequences=True, return\_state=True)  #训练模型时需要decoder的输出序列来与结果对比优化，故return\_sequences也要设为True  decoder\_output, \_, \_ = decoder(decoder\_input,initial\_state=encoder\_state)  #在训练阶段只需要用到decoder的输出序列，不需要用最终状态h.c  decoder\_dense = Dense(n\_output,activation='softmax')  decoder\_output = decoder\_dense(decoder\_output)  #输出序列经过全连接层得到结果    #生成的训练模型  model = Model([encoder\_input,decoder\_input],decoder\_output)  #第一个参数为训练模型的输入，包含了encoder和decoder的输入，第二个参数为模型的输出，包含了decoder的输出 |   2)用于推理的模型   |  | | --- | | #推理阶段，用于预测过程  #推断模型—encoder  encoder\_infer = Model(encoder\_input,encoder\_state)  #推断模型-decoder  decoder\_state\_input\_h = Input(shape=(n\_units,))  decoder\_state\_input\_c = Input(shape=(n\_units,))  decoder\_state\_input = [decoder\_state\_input\_h, decoder\_state\_input\_c]#上个时刻的状态h,c  decoder\_infer\_output,decoder\_infer\_state\_h,decoder\_infer\_state\_c=decoder(decoder\_input,initial\_state=decoder\_state\_input)  decoder\_infer\_state = [decoder\_infer\_state\_h, decoder\_infer\_state\_c]#当前时刻得到的状态  decoder\_infer\_output = decoder\_dense(decoder\_infer\_output)#当前时刻的输出  decoder\_infer=Model([decoder\_input]+decoder\_state\_input,[decoder\_infer\_output]+decoder\_infer\_state) |  5.3.3 encoder-decoder模型的搭建  |  | | --- | | def predict\_chinese(source,encoder\_inference, decoder\_inference, n\_steps, features):  #先通过推理encoder获得预测输入序列的隐状态  state = encoder\_inference.predict(source)  #第一个字符'\t',为起始标志  predict\_seq = np.zeros((1,1,features))  predict\_seq[0,0,target\_dict['\t']] = 1  output = ''  #开始对encoder获得的隐状态进行推理  #每次循环用上次预测的字符作为输入来预测下一次的字符，直到预测出了终止符  for i in range(n\_steps):#n\_steps为句子最大长度  #给decoder输入上一个时刻的h,c隐状态，以及上一次的预测字符predict\_seq  yhat,h,c = decoder\_inference.predict([predict\_seq]+state)  #注意，这里的yhat为Dense之后输出的结果，因此与h不同  char\_index = np.argmax(yhat[0,-1,:])  char = target\_dict\_reverse[char\_index]  output += char  state = [h,c]#本次状态做为下一次的初始状态继续传递  predict\_seq = np.zeros((1,1,features))  predict\_seq[0,0,char\_index] = 1  if char == '\n':#预测到了终止符则停下来  break  return output |  5.4关键问题 数据的预处理时由于数据量的过大，内存无法加载：可以通过修改默认的numpy的dtype来降低内存的需求。Numpy默认的np.dtype为float64,我们可以将其修改为float32，所需内存将减小一半。  原始数据集本有2万多条数据，但是由于数据量巨大，加上硬件条件的限制，导致训练的起来的速度非常的缓慢，所以仅仅将数据集的前10000条取出加以利用。  我们一共进行了200轮的训练，由于数据量的巨大和机器的性能的影响，我们训练模型的速度非常缓慢。 5.5 实验结果分析 图片1  图5-1程序运行结果图  如上图所示我们将取出的10000条数据中的前8000条作为训练集，后2000条作为测试集，一共进行了200轮的循环，8000条数据进行训练，实验运行结果如上图所示。我们可以看到在最后的时候损失和测试集的损失只有较小的浮动，证明该模型在此时已经趋于稳定，我们模型学习基本完成  图片2  图5-2实验效果图  我们可以从图5-2中看到，虽然输出的译文和我们原始的参考译文并不是完全一致，但是输出的译文流畅性较好，并且翻译出来的意思和原文的意思也基本一致。这是因为神经网络翻译模型并没有刻意去追求源语言和目标语言中词或短语在翻译时的一一对应，而是更加注重语言意义的整体的相似性。 参考文献  1. Hopfield J J.Neural networks and physical system with emergent collective computational abilities[J]. Proceedings of the National Academy of Sciences, 1982,79(8):2554-8. 2. Machine Learning Mastery:How to Develop an Encoder-Decoder Model for Sequence-to-Sequence Prediction in Keras(2018-11-25).[Online].Available URL:http://machine-Learnigmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/. 3. CSDN：对seq2seq的一些个人理解(2018-11-25).[Online].Available URL:https://blog.csdn.net/Zsaang/article/details/71516253. 4. CSDN：RNN与机器翻译（2018-11-25）.[Online].Available URL：<https://blog.csdn.net/alice>-yangxi1987/article/details/71055235. |