**实验四 神经机器翻译二**

**一、实验目的**

本次实验探讨了两个关键概念-子词（subword）建模和卷积网络-并将它们应用于我们在上一实验中构建的NMT系统。实验三 NMT模型可以看作四个阶段：

1. 嵌入层：将原始输入文本（源语句和目标语句）转换为词向量序列。

2．编码器：将源语句编码为编码器隐藏状态序列的RNN。

3．解码器：在目标语句上操作并借助编码器隐藏状态以产生解码器隐藏状态序列的RNN。

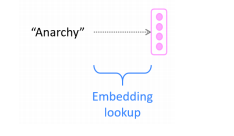
4. 输出预测层：具有softmax的线性层，在每个解码器时间步上产生下一个目标字的概率分布。

所有这四个子部分都在单词层次上模拟了NMT问题。在本实验的第一节中，我们将用基于字符的卷积编码器取代（1），在第二节中，我们将通过添加基于字符的LSTM解码器来增强（4）。这将有希望提高我们的BLEU测试集的性能！最后，在第三节中，我们将检查字符级编码器产生的字嵌入，并分析新的NMT系统的一些错误。

**二、模型描述**

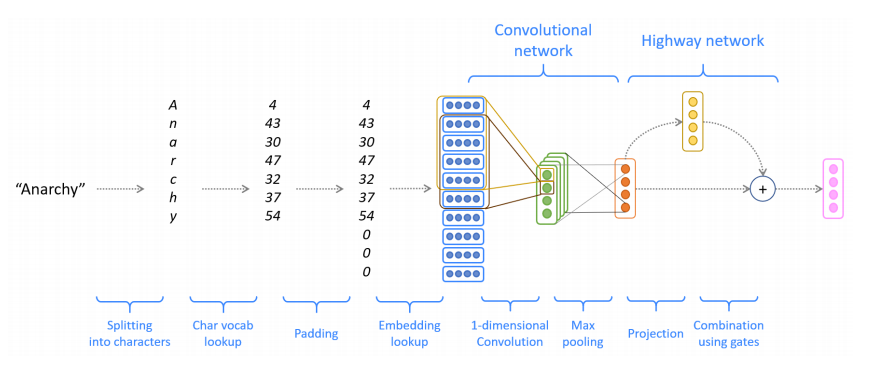
**1.字符级卷积编码器**

在实验三中，我们使用一个简单的查找方法来获得单词的表示。如果一个单词不在我们预先定义的词汇表中，那么它被表示为<UNK>标记（它有自己的嵌入）



**图1：实验三中基于查找的单词嵌入模型，它生成长度为的词嵌入。**

在本节中，我们将首先描述一种基于Kim等人在字符感知神经语言模型中的工作的方法，然后我们将实现它。具体来说，我们将用一系列更复杂的阶段替换图1中的“lookup”阶段，如图2所示：



**图2：基于字符的卷积编码器，最终生成长度为的词嵌入。**

图2中的模型有四个主要阶段，我们将以一个样本为例（而不是一个batch）描述这些阶段：

1. **将单词转换为字符索引**。我们有一个单词x(例如图2中的‘Anarchy’)。假设我们有一个预定义的字符表（例如，所有小写字母、大写字母、数字和一些标点符号）。通过查找每个字符的整数索引，我们可以将长度为l的单词x表示为整数向量：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\WKS1W1D)F0QP)YTJ(DC$80U.png

其中每个是字符表中的整数索引。

2. **填充和嵌入查找**。使用一个特殊的字符，我们将每一个单词填充（或截断），使其具有长度（定义为batch中最长单词的长度）。

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\JF21T9~6[V9CM})Z1}H}K4T.png

对于每个字符，我们查找一个字符嵌入（具有形状）。这将产生一个张量：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\RQCG03B%OSURJTZZW1ML}8U.png

在进入卷积网络之前，我们将对进行形状重塑以获得。

3.**卷积网络。**为了组合这些字符嵌入，我们将使用一维卷积。卷积层有两个超参数：卷积核大小（也称为窗口大小），它指示用于计算特征的窗口大小，以及过滤器的数量（也称为输出特征的数量或输出通道的数量）。卷积层有一个权重矩阵以及一个偏置向量。

为了计算第个输出特征（其中）, 对于输入的第t个窗口，在输入窗口以及权重矩阵之间执行卷积操作，并加上偏置项：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\@$5NJ~Z`JXTPC`I~S26U7PP.png

其中是两个形状相同的矩阵的元素相乘，sum是矩阵中所有元素的和。对每个特征和每个窗口执行此操作，其中。这些步骤会综合产生输出：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\}WN0L)MOE}(3ZP4XOK198TE.png

对于我们的应用程序，我们将设置为等于，即单词的最终的嵌入大小（图2中最右边的向量）。因此，

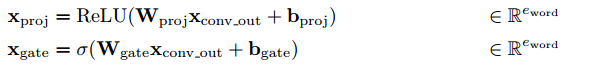


最后，我们将函数应用于，然后使用最大池化(max-pooling)将其转换为单个向量,这就是卷积网络的最终输出：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\7KDY5V%%V]PFBZB[U[E4~%D.png

在这里，取第二个维度上的最大值。给定矩阵，有。对于。

4**.Highway层和dropout。**Highway网络有一个由动态门控制的跳跃连接（skip-connection）。给定输入，我们计算：

`

其中权重矩阵，偏差向，是函数。接下来，我们使用门将一般投影和跳跃连接结合起来，获得输出

*C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\W86[6ATBD7K2287T$WJ3PIU.png*

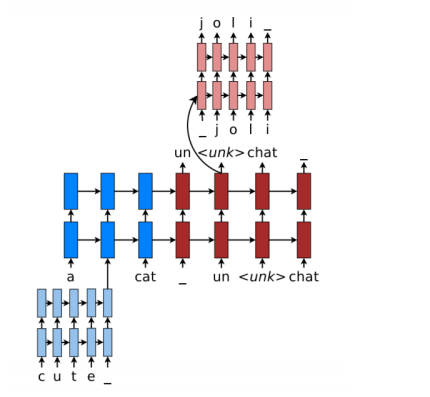
其中表示按元素进行的乘法运算。最后，我们向应用dropout：

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\~NC)[ZJK{%JI6$QV[V5OJZD.png

是我们将用于表示单词的嵌入——这将替换我们在实验三中使用的基于查找的单词嵌入。

**2.字符级LSTM解码器**

现在，我们将根据Luong&Manning的工作，在我们的NMT系统中添加一个基于LSTM的字符级解码器。其主要思想是，当我们的词级解码器生成一个<UNK>标记时，我们运行字符级解码器（您可以将其视为字符级条件语言模型）来每次生成目标单词一个字符，如图3所示。这将有助于我们产生生僻的和词汇外(oov)的目标词**。**

****

**图3：一个基于字符的解码器，如果基于单词的解码器生成UNK，则会触发。图由Luong&Manning提供。此图仅供参考，在我们的实现中，我们将只使用下面详细说明的一层LSTM。**

我们现在分三个部分描述模型：

1. **字符解码器的前向计算部分。**

给定一个整数序列代表一个字符序列，我们查找他们的字符嵌入,然后将其作为单向LSTM的输入。获得隐层状态以及单元状态:

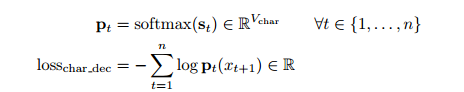
C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\04]T1CPQ[PUGS6V%E3}]]~Q.png

其中是CharDecoderLSTM隐层状态的大小。对于词级NMT解码器的当前时间步，初始的隐层状态，都被设置入组合输出向量(combined output vector)（参见实验三）。对于每个时间步我们计算分数（也称为logits）:

C:\Users\wfzw\Documents\Tencent Files\357971254\FileRecv\MobileFile\Image\U1J$F@PS_D`K~S735YM]%A2.png

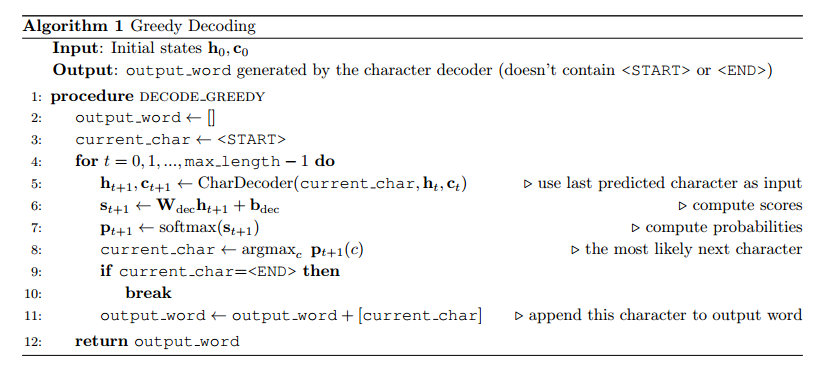
其中权重矩阵和偏置向量。如果我们使经过一个函数，我们将得到序列中下一个字符的概率分布。

**2.字符解码器的训练**。当我们训练NMT系统时，我们用目标句子中每个单词训练字符解码器（不仅仅是由<UNK>表示的单词）。例如，在主NMT解码器的特定步骤上，如果目标单词是music，那么CharDecoderLSTM的输入序列是，CharDecoderLSTM的目标序列是。我们将输入序列（以及从组合输出向量获得的初始状态）输入CharDecoderLSTM，从而获得分数，我们将其与目标序列进行比较。我们根据交叉熵损失之和进行优化：



注意，当我们计算一批单词的时，我们取整批单词的和（不是平均值）。在每次训练迭代中，我们将加入到基于词的解码器的损失中，从而同时训练基于词的模型和基于字符的解码器。

**3. 从字符解码器解码**。在测试时首先我们以通常的方式从我们的基于词的NMT系统产生翻译（例如，像beam search这样的解码算法）。如果翻译包含任何<UNK>标记，那么对于每个那样的位置，我们使用基于单词的解码器的组合输出向量来初始化CharDecoderLSTM的初始，然后使用CharDecoderLSTM来生成字符序列。为了生成字符序列，我们使用贪婪解码算法，该算法反复选择最可能的下一个字符，直到生成<END>标记或达到预定的最大长度。算法如下所示，仅针对一个样本（不成批）。



**三、实验要求**

**1.关于字符级编码器**

A．**（书面）**我们在课堂上学习到循环神经结构可以在可变长度输入下运行(即模型参数的形状与输入语句的长度无关。卷积架构也是这样吗？写一个句子来解释为什么或为什么不。

B.**（书面）**在基于字符嵌入的模型的第2步中，对于句子中的每个单词，我们将其填充到的长度。实际上，在我们的实现中，我们还为这个序列添加了两个特殊字符：一个字符在开头代表单词标记的开始，另一个字符在结尾代表单词标记的结束。因此，，其中为单词开头，为单词结尾。

在一维卷积中，我们做填充，即我们在输入的两边加上一些零，这样在输入上滑动的内核可以应用于至少一个完整的窗口。在这种情况下，如果使用内核大小k=5，我们使用一维卷积需要填充的大小是多少(即每边额外的零个数），以至于至少有一个窗口符合所有可能的值？解释你的理由。

提示：

•可能取得的最小值是多少？

•在1-d卷积层的输入中填充额外的零后（），更新后的的尺寸为 。

C.**(书面)**在步骤4中，我们引入了Highway网络，其中。既然是sigmoid的结果，他的范围是（0，1），考虑两种极端情形：当时；当时。这意味着Highway层在一般线性层（）一个简单地传递输入（）的层之间平滑地改变其表现。用一两句话来解释为什么这种行为在字符嵌入中很有用。根据的定义，你认为将初始化为负还是正比较好？简要解释你的理由

在第1节的剩余部分中，我们将在NMT中实现基于字符的编码器系统，尽管您可以在自己的实验3解决方案代码的基础上实现这一点，为了简单和公平，为你提供了一个基于实验3单词的NMT模型的完整实现（做了一些修改）；这是你完成实验四代码的基础。让我们从左到右实现图2中指定的整个网络。

D.**（编码）**实现vocab.py中的to\_input\_tensor\_char()方法

•使用vocab.py中的words2charindices()将所有单词中的每个字符转换为其在字符表中的相应索引；

•使用utils.py中的pad\_sents\_char()方法将所有单词填充到该batch中所有单词的最大单词长度，并将所有句子填充到该batch中所有句子的最大句子长度。

然后将得到的填充语句转换为张量。这对应于图2的前三个步骤（分割、vocab查找和填充）。确保你进行了reshape使得你的输出有以下形状：（max\_sentence\_length,batch\_size,max\_word\_length）

E．**（编码）**在highway.py空文件中，实现highway网络，将highway作为nn.Module的一个名为Highway的类。

•你的模型应该包括一个\_\_init\_\_()方法和一个forward()方法（输入输出由你自己决定）

•forward()方法应该将映射到

•注意，尽管上面的模型描述不是批处理的，但是forward（）函数应该操作成批的单词。

•确保你的模块使用两个nn.Linear的层

F.**（编码）**在cnn.py空文件中，实现卷积网络网络,将卷积网络作为nn.Module的一个名为CNN的类。

•你的模型应该包括一个\_\_init\_\_()方法和一个forward()方法（输入输出由你自己决定）

•forward()方法应该将映射到

•注意，尽管上面的模型描述不是批处理的，但是forward()函数应该操作成批的单词。

•确保你的模块具有一个nn.Conv1d类型的实例。

•卷积核大小设置为k=5，填充长度padding=1。

**G. (编码)** 在model\_embedding.py中为ModelEmbeddings类编写\_\_init\_\_()和forward()

•forward()函数必须从映射到–请注意，这是针对整批句子，而不是成批单词。

•你将会使用你创造的CNN和Highway模块；

•不要忘记dropout层，使用0.3的dropout概率

•你的ModelEmbeddings类需要包含一个nn.Embedding对象，它还应该包含一个名为self.word\_embed\_size的字段。

•我们将使用

•根据你对CNN和Highway的实现,你可能需要对张量形状进行重塑以使其达到CNN和Highway所要求的形状，然后再重新构造以获得ModelEmbeddings.forward()的最终输出。

**H. （编码）**在nmt\_model.py中，完成forward()方法从而将词级的编码替换为字符级填充后的编码。这将你的ModelEmbedding代码与你编写的预处理代码联系在一起。

**I．（编码）**在你的机器上执行sh run.sh train\_local\_q1训练模型，并执行sh run.sh test\_local\_q1测试它。在训练中，你应该观察到平均损失降到接近0，一旦你运行测试，你应该观察到BLEU分数在测试集上高于99.00. 如果不观察这些情况，可能需要debug。重要提示：请确保不要修改由代码生成的输出文件(outputs/test\_outputs\_local\_q1.txt)

1. **关于字符级解码器**
2. **（编码）**在char\_decoder.py中编写forward()函数。此函数接受输入和（,）（在字符解码器正向计算部分描述）并返回和（,）。
3. **（编码）**在char\_decoder.py中编写train\_forward()函数。这个函数计算整个batch的求和。提示，仔细阅读nn.CrossEntropyLoss文档
4. **（编码）**在char\_decoder.py中编写decode\_greedy()函数。以实现贪婪解码（DECODE\_GREEDY）算法。注意你要在batch上实现算法1，尽管上面是按一个样本来描述的。算法1表明你要在遇到<END>标记时终止算法。但在批处理的情况下，您可能会发现完成for循环的所有max\_length步骤，然后截断out\_words更为方便。
5. **（编码）**在你的机器上执行sh run.sh train\_local\_q2训练模型，并执行sh run.sh test\_local\_q2测试它。在训练中，你应该观察到平均损失降到接近0，一旦你运行测试，你应该观察到BLEU分数在测试集上高于99.00. 如果不观察这些情况，可能需要debug。重要提示：请确保不要修改由代码生成的输出文件(outputs/test\_outputs\_local\_q2.txt)
6. 现在我们已经实现了基于字符的编码器和基于字符的解码器，现在终于到了训练整个系统的时候了！执行sh run.sh train训练你的模型，在测试集上运行你的模型sh run.sh test。并在作业报告中报告你的BLEU得分。还要确保outputs/test\_outputs.txt提交且未修改。我们将使用它来验证您报告BLEU分数。

**四、实验提交及截止日期**

本次实验你需要提交三分材料(1)代码及模型部分（2）书面分析部分（3）实验报告。

**•** 将完整的代码文件、训练保存的模型等文件打包为”实验四代码部分”

**•** 一份书面分析材料，回答实验要求中有关书面分析的问题。

**•** 一份实验报告，记录你是如何实现NMT系统，可对方法设计及编码进行解释说明。

**•** 请将上述需要提交的材料打包，文件命名格式为姓名-学号-实验四.zip（rar），于6月5日晚24时之前[发送到357971254@qq.com](mailto:发送到357971254@qq.com)