计算语言学作业6

陈禹东 计研三二 2018210965

一、实验内容

1. 实验内容：

基于RNN语言模型，计算句子的概率。

实验测试不同LSTM隐状态大小（hidden\_size）超参数下的结果。

实验数据（北大《人民日报》标注语料库）分为训练集、验证集和测试集，验证集、测试集用于计算所有句子的perplexity的平均值（验证集用于确定最佳超参数）。

其中为一个长度为N的句子。

2. 实验设定：

(1) 分句方式：按句末标点(。！？)分句，并将句末标点用<eos>代替（句中的标点符号也算一个词）。

(2) 单词处理：删去了与某个词相连的标点，如“[新华社”处理为“新华社”。不保留词性信息，不同词性的同一词汇看作一个词。

(3) 基于tensorflow神经网络框架：tensorflow1.5.0+cuda9.0+cudnn10.0。显卡：GeForce GTX 660 Ti。

(4) 参考文档：<https://www.tensorflow.org/tutorials/sequences/recurrent>

代码：<https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb>

二、实验步骤与原理

1. 数据预处理

首先，依据训练集语料构建单词库。

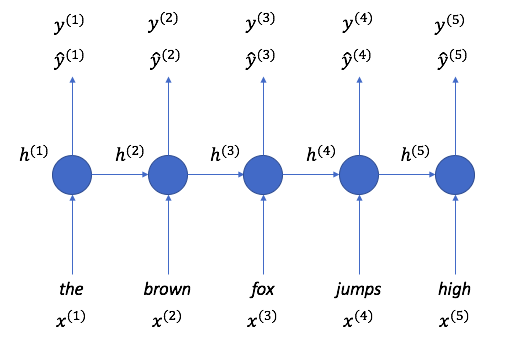
训练集里一共有49003个不同单词，为了减少神经网络的参数，缩短训练时间，我们将频次<6的单词替换为<unk>（数量占比7.3%），剩下11250个不同单词，作为单词库（于是vocab\_size = 11250）。

然后，对于验证集和测试集，我们将频次<6或未在单词库中出现的单词都替换为<unk>。

由三个数据集分别生成三个.txt文件，文件的每一行包含一个用空格分词的句子。

2. 神经网络的输入和label

为使RNN的训练过程更加可控（Back Propagation更易于计算），本实验采用截断式BP的static RNN（unrolled RNN），也就是说，LSTM的输入数和输出数始终是固定的num\_step个。不妨设num\_step=5。



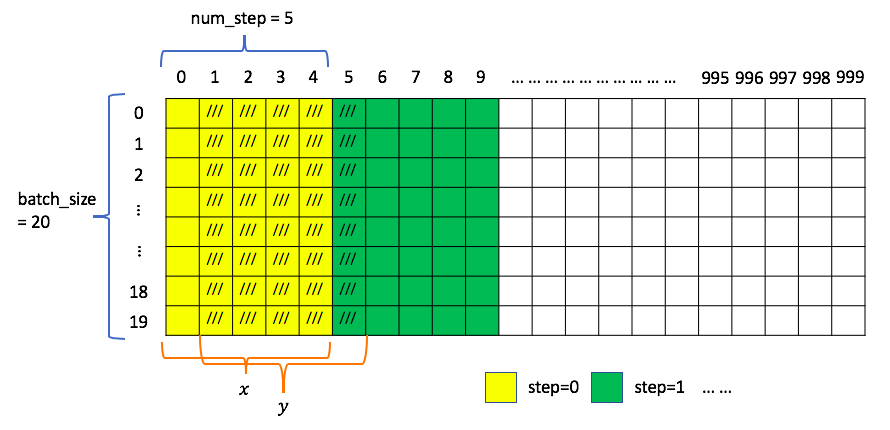
*Static RNN 示意图（num\_step=5）*

于是，训练的大致过程就是：每轮训练（epoch）的每一步（step）前向传播过程中，我们都需要向LSTM投喂num\_step=5个单词作为输入，得到5个预测输出（每投喂一个单词，预测下一个单词在词库所有单词上的softmax概率），并对应地提供5个label与预测输出作比对，计算loss，再通过BPTT（Back Propagation Through Time）计算梯度，并更新网络参数。

考虑到minibatch训练策略（不妨设batch\_size=20），我们每个step则需要投喂batch\_size\*num\_step=20\*5个单词作为输入，并对应提供20\*5个label。

为了满足这个需要，我们需要对预处理好的.txt数据稍作变换。

首先，我们将所有的换行符用<eos>替代，并把词库里的所有单词用正整数id编号，把.txt里的单词全部转换为对应的id。然后把所有句子依次相连，得到长度为data\_len（不妨设=20000）的一长串单词串。然后，我们将它等分为batch\_size =20段，每段有batch\_len=20000/20=1000个单词。对于每一段，我们再将它等分为长度=num\_step=5的很多个小段。于是，如图所示，第一步前向传播要投喂的，就是黄色区域的20\*5个单词，斜线区域为20\*5个label。



*数据集变换与单词投喂示意图*

第二步前向传播时，则使用第二个20\*5的单词矩阵作为输入，以此类推，直到把数据集的所有单词都过一遍，完成一整轮训练。

值得注意的是，我们投喂时并不是直接投喂单词本身，而是需要事先将它们转换为相应的embedding词向量。词向量的维数与隐状态的维数hidden\_size（不妨设=200）一致，这是为了让LSTM中与维数对齐。Word2vec的转换用到了一个vocab\_size \* hidden\_size = 11250 \* 200的映射矩阵，即词库里的每一个单词对应一个200维的向量。这个矩阵是可训练的。

这样，LSTM单元每接收20\*5个单词，实际的输入shape是(20, 5, 200)，输出的shape也是(20, 5, 200)。

3. Loss与perplexity

由上述过程可以知道，每轮训练的每一步，我们依次向LSTM单元投喂20\*5个200维的词向量（每个时间点投喂20\*1个，依次喂完20个长度为5的“句子”），得到了20\*5个200维的预测结果output。为了将output与20\*5个label作比对，我们先用一个200\*11250的全连接层（Wx+b），将output的shape变为(20, 5, 11250)，可理解为此时的output包含了20\*5个11250维的向量, 。与之对应有20\*5个label（正整数单词id）。

下面考虑第0个向量和第0个label ，要如何计算它们之间的loss。

对于，我们将其做一个softmax变换，得到的新向量每一维代表着模型预测“下一个单词是词库中第i号单词”的概率。

对于，假设，表示它是词库里的第3297号单词，那么可以把它扩展成一个11250维的0-1向量，使其第3297维为1，其余都为0。则此时和如下所示：

下标： 0 1 … … 3297 … … 11248 11249

(q): [0.0003, 0.0002, … …, 0.2, … …, 0.00001, 0.00007]

(p): [ 0, 0, … …, 1, …. …, 0, 0]

我们可把视作待预测单词（一个离散型随机变量X，取值范围为[0, 11249]）的真实分布p，则视作模型对它的预测分布q。我们的目标是使得q尽可能逼近p，于是我们需要让p与q的KL散度（衡量两个分布间的距离）最小：

其中是X在真实分布p之下的熵（为一个确定值），为两个分布的交叉熵——于是转化为需要让交叉熵最小。因此，我们定义的Loss也就等于minibatch上所有20\*5个单词的交叉熵的平均值。注意到当且仅当（如上例中的3297）时为1，取其它值时为0，因此有（N=20\*5）：

这和参考文档中的Loss公式一致。

那么，这个Loss与我们要研究的困惑度Perplexity有什么联系呢？官方文档给出的答案是：。事实上这是容易证明的：

由于困惑度本身是用于评价语言模型的，因此这里使用了模型的预测分布q。另外，倒数第二个等号是因为在LSTM这种循环神经网络模型里，第n个时间点输出的都是之前n-1个时间点的单词输入为已知信息的，因此事实上是一个条件概率——这样，就与作业要求里句子概率计算的定义完全一致了。

值得一提的是，上述的N是minibatch上所有单词的总数20\*5，而普通困惑度定义中的N是一个句子S的长度，似乎并不匹配。但在之前的N-gram和HMM实验中，我们计算的验证集或测试集上的Perplexity值为该数据集所有句子的困惑度的平均值，而本次实验计算的相当于是batch\_size=20个句子困惑度的平均值。因此，本次实验得到的Perplexity是可以与之前实验直接做比较的。

二、实验结果和分析

1. 实验结果：

|  |  |  |
| --- | --- | --- |
| Model | Valid set | Test set |
| Markov (uni/bigram)1 | 10257.83/9313.86 | 7508.20/9048.78 |
| Markov (uni/bigram)2 | 4710.67/975.67 | 3737.06/88.16 |
| HMM (forward) | 438.82 | 434.71 |
| HMM (backward) | 438.82 | 434.71 |
| LSTM (50) | 104.95 | 86.40 |
| LSTM (100) | 96.51 | 78.82 |
| LSTM (200) | 96.56 | 78.13 |
| LSTM (500)1 | 121.93 | 95.32 |
| LSTM (500)2 | **72.33** | **58.52** |

Markov1、HMM平滑方法：Adding one smoothing

Markov2平滑方法：Good Turing

LSTM (hidden\_size)：括号里代表隐状态的维数

LSTM (50) (100) (200) 和 (500)1：除了hidden\_size以外，使用了官方代码中small的超参配置

LSTM (500)2：除了hidden\_size以外，使用了官方代码中medium的超参配置

2. 实验分析：

首先，perplexity越小，句子的概率越大（perplexity的定义已消除句子长度的差异）。而在其他条件都相同的情况下，对于同一个句子，若采用方法A和方法B估计的句子概率为PA和PB，且PA>PB，则可以认为A是一个更好的估计方法，它让我们对句子的估计更加确定。

基于这个原则，我们可以发现：

(1) LSTM模型相比之前的n-gram和HMM模型，能更好地估计句子的概率。尽管它没有用到词性信息，但它庞大的参数集和多轮的迭代训练让它具备了强大的刻画能力，以致于可以很好地学习出句子概率的估计模式。

(2) LSTM模型相当于一个精细训练过的n-gram（实验中n=num\_step=20）模型，只是它的参数不再基于训练集上的贝叶斯统计，而是使用LSTM模型的模型参数。它使用的参数量远小于相同n的n-gram模型，但它对条件概率的估计却也能利用这有限的参数量（空间开销）达到很好的效果。与此同时，LSTM模型的训练时间开销相应地会远大于n-gram模型。

关于参数规模，我们可以作如下估算。对于n=20的n-gram模型，我们需要计算出所有可能的条件概率：。假设词库单词总数为10000 = 104，则上述第i组条件概率对应的参数总量为，20组条件概率的参数总和为，参数量已爆炸，且十分稀疏。而对于n=num\_step=20、hidden\_size=200的LSTM模型，打印tf.trainable\_variables()可知，其主要参数量在word2vec矩阵和LSTM输出后接的全连接层，参数量略大于。

(3) 观察实验结果可知，在其他超参都与之匹配时，hidden\_size越大，迭代收敛后的Perplexity越小，模型估计句子概率的效果越好。但前提是其他超参与之匹配，如LSTM (500)1 的learning rate减小太快，迭代次数又太少，因此效果反而不如LSTM (50)。

(4) 本次实验中为了防止训练时间太长，我对频次小于threshold=6的单词都标记为了unk，这是与之前的实验不同之处。由于时间关系，没能对不同的threshold做对比实验，但我猜测threshold越小，即词库单词数越大时，LSTM需要预测的小概率更多，相对误差会变大，从而Perplexity也会变大。

3. 实验感悟

本次实验花了较长的时间仔细钻研了LSTM模型的数据流和训练过程，并逐渐熟悉了TensorFlow神经网络框架的特点，收获很大。

感谢老师和助教的启发和帮助！