基于注意力机制和循环神经网络的语言模型

构建语言模型是自然语言处理中一项重要的任务。其构建方法基本可分为两种，一种是：专家语法规则模型，另一种是：利用统计原理原理构建语言模型。在现阶段人们的探究中，后者基本为人们所常用，搭建出来的语言模型实验效果也较好。因此，在本次作业中，我们基于注意力机制和循环神经来构建语言模型，并行进行对其进行实验和分析。

背景知识

统计语言模型就是计算一个句子的概率大小的这种模型。形式化讲，统计语言模型的作用是为一个长度为m的字符串确定一个概率分布：

$$P(w_1, w_2, ..., w_n)$$

表示其存在的可能性，其中w_1到w_n依次表示这段文本中的各个词。计算机借助概率语言模型可以估计出自然语言中每个句子出现的可能性，而不是简单的判断该句子是否符合文法。常用统计语言模型，包括了N元文法模型$N-gramModel$)统计语言模型把语言（词的序列）看作一个随机事件，并赋予相应的概率来描述其属于某种语言集合的可能性。给定一个词汇集合 V，对于一个由V中的词构成的序列$S=\{w_1, w_2, ..., w_m\}\epsilon V$，统计语言模型赋予这个序列一个概率$P(S)$，来衡量$S$符合自然语言的语法和语义规则的置信度。

在此，把词设为最小的结构单位，并且设S由词$w_1, w_2, ..., w_n$组成，那么，不失一般性，$P(S)$可由下面公式计算：

$$P(S)=P(w_1)P(w_2|w_1)P(w_3|w_1,w_2).....P(w_n|w_1,w_2,.....,w_n)=\prod_{i=1}^{n}P(w_n|w_i,w_2,.....,w_{i-1})$$

不过，由于当n较大时，n元组$w_1w_2.....w_n$所在的数据空间将会十分巨大，空间中元素个数随n呈指数增长，这对$w_1w_2.....w_n$的统计将会很不充分，这样我们也不能得到$P(S)$的准确估计。这也就是所谓的数据稀疏问题。为了解决这个问题，在t元语法模型中，我们假设任意一个语法单元w_n仅和它前面的$t-1$个语法单元相关。即:

$$P(w_n|w_1,.....,w_{n-1})\approx P(w_n|w_{n-t+1},.....,w_{n-1})$$

这样，式就转化为:

$$P(S)=P(w_1).....P(w_n|w_{n-t+1}.....w_{n-1})$$

由于$w_{n-t+1},.....,w_{n-1}$要比$w_1,.....,w_{n-1}$的数据充分。这样只要选择合理的t再加上一些平滑技术（smoothing techniques），对$P(w_n|w_{n-t+1}.....w_{n-1})$的估计还是比较准确的。组合这些也可以得$P(w_n|w_{n-t+1}.....w_{n-1})$到相对准确的$P(S)$。通常，在统计机器翻译中，$t=3$或$t=4$就可以得到令人满意的结果。从本质上说，$t$元语法模型把所有以$w_{n-t+1},.....,w_{n-1}$结尾的$w_1,.....,w_{n-1}$聚类到了一起。这样也就缓解了$w_1,.....,w_{n-1}$的稀疏性。

统计语言模型用简单的方式，加上大量的语料，产生了比较好的效果。统计语言模型通过对句子的概率分布进行建模，统计来说，概率高的语句比概率低的语句更为合理。在实现中，通过给定的上文来预测句子的下一个词，如果预测的词和下一个词是一致（该词在上文的前提下出现的概率比其它词概率要高），那么上文+该词出现的概率就会比上文+其他词的概率要更大，上文+该词更为合理。

任务简介

在本作业中，我所定任务是：搭建语言模型，并且用已有的语料库来训练所搭建的语言模型，最后实现所训练的语言模型可以根据句子的上文来预测下一个词。

其中，对搭建的语言模型我们会做出前向传播、反向传播、使用注意力机制来模型优化、超参数调节和结果分析等具体详细的工作。

编程工作：

- 对数据进行预处理、批处理等操作。
- 实现语言模型。
- 添加注意力机制。

实验数据集

在实验中，我使用Penn Treebank（PTB）数据集，其是一个项目的名称，项目目的是对语料进行标注，标注内容包括词性标注以及句法分析。PTB数据集在自然语言任务中使用广泛，而且相对而言其里面的噪声数据较少，真实度较高，所在在实际训练、验证和测试的过程中可信度相对而言较高。因此，我们选用PTB数据作为我实验的训练、验证和测试的数据。

模型的评价指标

在本作业中，我们使用困惑度（perplexity，PPL）来评价模型的好坏，PPL是用在自然语言处理领域（NLP）中，衡量语言模型好坏的指标。它主要是根据每个词来估计一句话出现的概率，并用句子长度做normalize，公式为：

$$PP(S)=P(w_1,w_2,.....,w_N)^{-N}=\sqrt[N]{\prod_{i=1}^{N}\frac{1}{p(w_i|w_1,w_2,.....,w_{i-1})}}$$

根据公式可以看出：模型的PPL得值越小证明模型越优，所以在实验的过程，我们的目标就是把语言模型的PPL值尽量降到最低值。

其中使用PPL来衡量语言模型的好坏的原因有以下：

- 训练数据集越大，PPL会下降得更低，1billion dataset和10万dataset训练效果是很不一样的；
- 数据中的标点会对模型的PPL产生很大影响，一个句号能让PPL波动几十，标点的预测总是不稳定；
- 预测语句中的“的，了”等词也对PPL有很大影响，可能“我借你的书”比“我借你书”的指标值小几十，但从语义上分析有没有这些停用词并不能完全代 表句子生成的好坏。

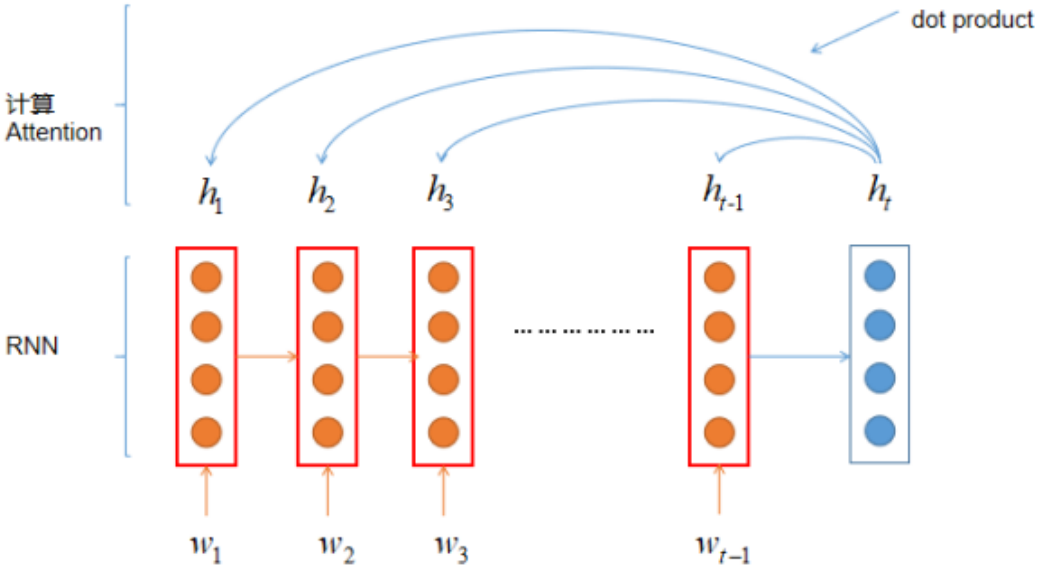
模型构建

单一的用RNN来搭建语言模型，可以看出结果虽然可观，但是仍旧有较大的改进和提升的空间，在本章节中，我们使用注意力机制和更换优化器两方面来对已经构建的语言模型进行改进，提升性能和衡量指标的PPL。

Attention机制（注意力机制）的原理就是计算当前输入序列与输出向量的匹配程度，匹配度高也就是注意力集中点其相对的得分越高，其中Attention计算得到的匹配度权重，只限于当前序列对，不是像网络模型权重这样的整体权重。

同理，在下一词预测的时候，我们前面搭建的语言模型，其实默认已有词w_1、w_2、.....、w_{t-1}对预测词w_t影响是一样的。但实际上在很多句子中w_1、w_2、.....、w_{t-1}对w_t的影响并不是一样的，例如：It will rain today, so you should take an umbrella. 当我们预测umbrella一词的时候，显然rain一词对其影响的程度最大，所以应该赋予它较高的注意力。

从而构建语言模型的注意力机制如下图所示：



如上图所示，我们采用h_t和分别前面时刻神经元状态h_t进行点积的方法求解注意力分数$Score_i$，即得到Attention分数向量为：

$$AttentionScore = [s_t^T, ..., s_t^T h_{t-1}]$$

再利用$softmax$函数将$AttetionScore$转化概率分布：

$$a^t = softmax(AttentionScore) \epsilon R^N$$

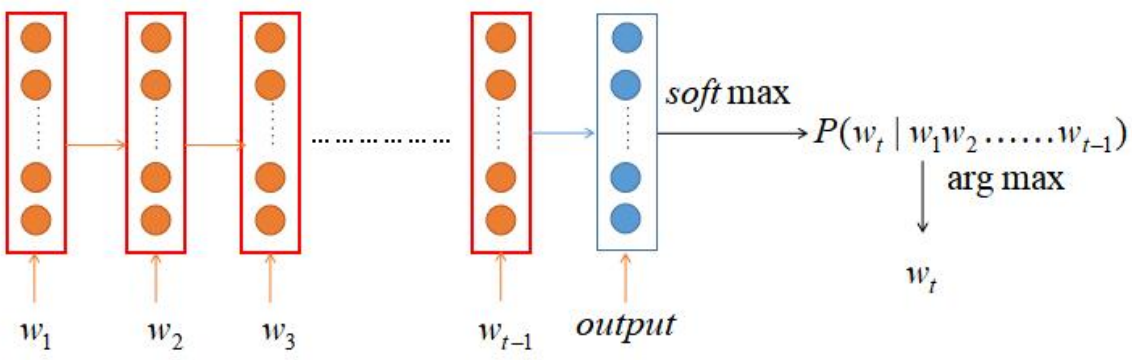
从而在计算$h_1, ..., h_{t-1}$对预测词的影响力的时候，可以利用以下公式：

$$a_t = \sum_{i=1}^{N} a_i^t h_t \epsilon R^h$$

所以，在最后输出层变换为如下：

$$\tilde{y} = [a_t; h_t]W + b$$

模型总的框架图如下所示：



此部分关键代码为：

```
attention = []
for it in outputs[:-1]:
    attention.append(torch.mul(it, output).sum(dim=1).tolist()) #get attention s
core
attention = torch.tensor(attention)
attention = attention.transpose(0, 1)
attention = nn.functional.softmax(attention, dim=1).transpose(0, 1)
#get soft attention
attention_output = torch.zeros(outputs.size()[1], n_hidden)
for i in range(outputs.size()[0]-1):
  attention_output += torch.mul(attention[i],outputs[i].transpose(0,1)).transpos
e(0,1)
output = torch.cat((attention_output, output), 1)
#joint ouput output:[batch_size, 2*n_hidden]
```

完成的实现代码已经开源，开源项目的链接为https://github.com/Chenglong-coder/MTcourse/tree/main/Project。

反向传播参数优化过程改进

在我们的任务中，由于参数较多，而且训练数据较为庞大，经过分析，在反向传播过程中，简单的梯度下降法已经难以任务的学习过程，所以在此我们采用了Adam学习方法来进行改进参数学习过程。

Adam优化器结合**AdaGrad**和**RMSProp**两种优化算法的优点。对梯度的一阶矩估计（**First Moment Estimation**，即梯度的均值）和二阶矩估计（**SecondMoment Estimation**，即梯度的未中心化的方差）进行综合考虑，计算出更新步长。

主要包含以下几个显著的优点：

- 实现简单，计算高效，对内存需求少；
- 参数的更新不受梯度的伸缩变换影响；
- 超参数具有很好的解释性，且通常无需调整或仅需很少的微调；
- 更新的步长能够被限制在大致的范围内（初始学习率）；
- 能自然地实现步长退火过程（自动调整学习率）；
- 很适合应用于大规模的数据及参数的场景；
- 适用于不稳定目标函数；

- 适用于梯度稀疏或梯度存在很大噪声的问题。

  具体的Adam参数更新规则如下：

  首先计算t时间步的梯度：

$$g_t = \nabla_\theta J(\theta_{t-1})$$

然后，计算梯度的指数移动平均数，m_0初始化为0。为了是的更新的较为准确，从而综合考虑之前时间步的梯度动量。β_1系数为指数衰减率，控制权重分配（动量与当前梯度），通常取接近于1的值。默认为0.9。

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

其次，计算梯度平方的指数移动平均数，v_0初始化为0。β_2系数为指数衰减率，控制之前的梯度平方的影响情况。类似于RMSProp算法，对梯度平方进行加权均值。默认为0.999。

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2$$

之后，由于m_0初始化为0，会导致m_t偏向于0，尤其在训练初期阶段。所以，此处需要对梯度均值mt进行偏差纠正，降低偏差对训练初期的影响。

$$\tilde{m}_t = \frac{m_t}{1-\beta_1^t}$$

与m_0类似，因为v_0初始化为0导致训练初始阶段v_t偏向0，对其进行纠正。

$$\hat{v} = \frac{v_t}{1-\beta_2^t}$$

对于更新参数，初始的学习率α乘以梯度均值与梯度方差的平方根之比。其中默认学习率$\alpha = 0.001$，$\varepsilon = 10^{-8}$避免除数变为0。由表达式可以看出，对更新的步长计算，能够从梯度均值及梯度平方两个角度进行自适应地调节，而不是直接由当前梯度决定。最终更新参数的公式如下：

$$\theta_t = \theta_{t-1} - \frac{\alpha^* \tilde{m}_t}{\sqrt{\tilde{v_t}} + \varepsilon}$$

实验结果及分析

为了更好的对比优化现象，我们分别对无优化语言模型（RNNLM）、带注意力机制语言模型（RNNLM-Attention）、Adam优化器更新（RNNLM-Adam）和带注意力机制而且使用Adam更新的语言模型（RNNLM-Attention-Adam）分别进行实验来进行对比分析。

在这里，我们为了保证对比的效果明显，我们在这几个改进的语言模型里面设置的超参数都相同，分别:上义词的个数embedding_size = 3，训练的时候批的大小batch_size = 256，学习率的大小learn_rate = 0.001。实验的结果如下：

Model	Training Time(h)	PPL(train)	PPL(valid)	PPL(test)
RNNLM	10.5	116	135	127
RNNLM-Attention	15.6	104	115	110
RNNLM-Adam	***9.6***	112	133	125
RNNLM-Attention-Adam	14.6	**100**	***110***	***103***

在上表中，我们分别记录了每次实验中，训练集、验证集和测试集的PPL值，还有训练到收敛所需要的时间单位为小时。从表中的实验结果数据来看：RNNLM-Attention-Adam模型的实验结果是最好的，另外可以对比RNNLM-Adam和RNNLM模型的实验数据来看，Adam优化器相比普通的梯度下降优化的方法，对PPL的影响较小，但是相对而言缩短了训练的时间。而且，对比RNNLM-Attention和RNNLM模型的数据来看，Attention机制对实验的结果有明显提升作用，但是由于其中需要计算Attention分数和Attention矩阵导致的训练的时间有所增加。

实验环境

在实验中，我们采用Python3.6进行编码，来实现我们的模型设计和测试。由于数据集的庞大，我们在服务器上运行我们的程序，其中运行环境信息为：

- 运行服务器的CPU型号为：Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz
- 服务器的CPU的个数为：2
- 服务器内核信息为：Linux IP-219-216-64-129.neu.edu.cn 3.10.0-957.21.2.el7.x86_64 #1 SMP Wed Jun 5 14:26:44 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
- 服务器内存信息为：total->62G used->16G

配置需求：

- OS: Linux or Windows
- Python>=3.6
- CUDA >= 9.2, <= 10.0 (optional)
- PyTorch>=1.10
- NumPy>=1.18

程序说明

运行命令：

```
python RNN_attention_torch.py \
-train $trainFile \
-valid $validFile \
-test $testFile \
-n_hidden $hiddenNumber \
-train_batch_size $trainBatchSize \
-valid_batch_size $validBatchSize\
-num_layers $layerNumber \
-size_window $windowSize \
-learn_rate $learnRate
```

参数说明：

- train -训练集的路径。自定义为：'./data/train.txt'。
- valid -验证集的路径。自定义为：'./data/valid.txt'。
- test -测试集的路径。自定义为：'./data/test.txt'。
- n_hidden -隐藏层的大小。自定义为：5。
- train_batch_size -训练的时候batch的大小。自定义为：512。
- valid_batch_size -验证的时候batch的大小。自定义为：512。
- num_layers -RNN的层数。自定义为：1。
- size_window -窗口的大小。自定义为：3。
- learn_rate -学习率大小。自定义为：0.0001。

个人总结

在《语言分析和机器翻译》这门课的学习过程中，由于之前在实验室学习过一段时间，所以对课程的相关知识稍微有一些了解，但是通过这门课程的学习，我对语言分析和机器翻译有了更为系统的学习。并且，在课上，通过听老师的讲解，也对我一些对此课程相关的一些疑问得到了解答和启发。除此之外，在本课程的学习过程中，我也学习到了很多的前沿知识，例如神经网络，反向传播等等。这些知识可以说是对很多领域都是通用的，不仅仅只限于对语言的分析上，这也为我接下来的研究扩宽了视野。

在课程的学习过程中，朱老师还过来给我们讲述了一些科研及创业道路上的一些历程和心得。在我看来，这些心得都是很珍贵的，听完之后，对我的情商或者是意志力也是一种提升的。

为了更好的学习课程，我也在课下之余对实验室出版的书《机器翻译：统计建模与深度学习方法》进行了阅读。书籍写的简单易懂，生动有趣，把很多比较复杂的问题，进行活泼有趣的讲述出来了。对我印象最深的就是讲述感知机的那块，书籍用一个生动有趣看电影的例子去简述了感知机的形成及内部机制，并且也引出了深度学习的一系列研究的点，很是绝妙。

在最后，感谢肖老师和其他任课老师在课上悉心地讲述课程知识，让我有机会去提升自己，培养自己的能力，接触到较为前沿的知识，为以后的学习和实践打下了坚实的基础，受益匪浅。

反馈

项目仅供学习使用，欢迎交流！联系邮箱：chenglong1119@qq.com