1 Sep

玩转Keras之seq2seq自动生成标题

By 苏剑林 | 2018-09-01 | 354962位读者 引用

话说自称搞了这么久的NLP，我都还没有真正跑过NLP与深度学习结合的经典之作——seq2seq。这两天兴致来了，决定学习并实践一番seq2seq，当然最后少不了Keras实现了。

seq2seq可以做的事情非常多，我这挑选的是比较简单的根据文章内容生成标题（中文），也可以理解为自动摘要的一种。选择这个任务主要是因为“文章-标题”这样的语料对比较好找，能快速实验一下。

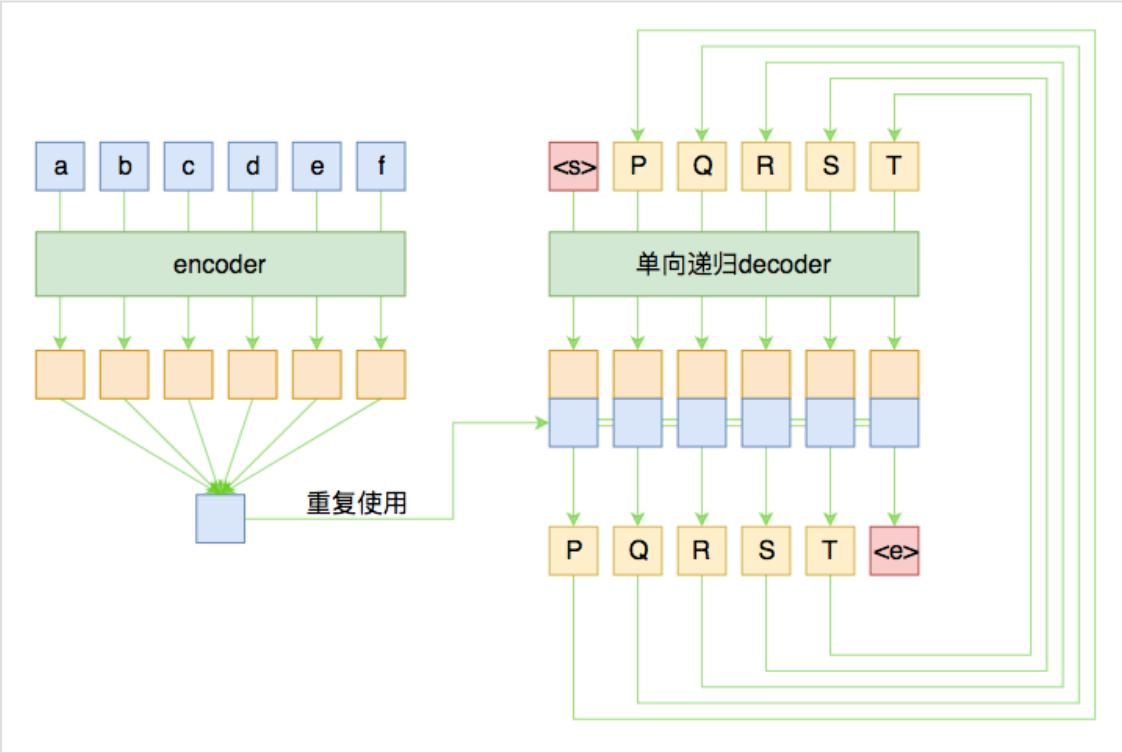
seq2seq简介

所谓seq2seq，就是指一般的序列到序列的转换任务，比如机器翻译、自动文摘等等，这种任务的特点是输入序列和输出序列是不对齐的，如果对齐的话，那么我们称之为序列标注，这就比seq2seq简单很多了。所以尽管序列标注任务也可以理解为序列到序列的转换，但我们在谈到seq2seq时，一般不包含序列标注。

要自己实现seq2seq，关键是搞懂seq2seq的原理和架构，一旦弄清楚了，其实不管哪个框架实现起来都不复杂。早期有一个第三方实现的Keras的seq2seq库，现在作者也已经放弃更新了，也许就是觉得这么简单的事情没必要再建一个库了吧。可以参考的资料还有去年Keras官方博客中写的《A ten-minute introduction to sequence-to-sequence learning in Keras》。

基本结构

假如原句子为$X=(a,b,c,d,e,f)$，目标输出为$Y=(P,Q,R,S,T)$，那么一个基本的seq2seq就如下图所示。



基本的seq2seq架构

尽管整个图的线条比较多，可能有点眼花，但其实结构很简单。左边是对输入的encoder，它负责把输入（可能是变长的）编码为一个固定大小的向量，这个可选择的模型就很多了，用GRU、LSTM等RNN结构或者CNN

+Pooling、Google的纯Attention等都可以，这个固定大小的向量，理论上就包含了输入句子的全部信息。

而decoder负责将刚才我们编码出来的向量解码为我们期望的输出。与encoder不同，我们在图上强调decoder是“单向递归”的，因为解码过程是递归进行的，具体流程为：

> 1、所有输出端，都以一个通用的<start>标记开头，以<end>标记结尾，这两个标记也视为一个词/字；
>
> 2、将<start>输入decoder，然后得到隐藏层向量，将这个向量与encoder的输出混合，然后送入一个分类器，分类器的结果应当输出P；
>
> 3、将P输入decoder，得到新的隐藏层向量，再次与encoder的输出混合，送入分类器，分类器应输出Q；
>
> 4、依此递归，直到分类器的结果输出<end>。

这就是一个基本的seq2seq模型的解码过程，在解码的过程中，将每步的解码结果送入到下一步中去，直到输出<end>位置。

训练过程

事实上，上图也表明了一般的seq2seq的训练过程。由于训练的时候我们有标注数据对，因此我们能提前预知decoder每一步的输入和输出，因此整个结果实际上是“输入X和$Y_{[:-1]}$，预测$Y_{[1:]}$，即将目标Y错开一位来训练。这种训练方式，称之为Teacher-Forcing。

而decoder同样可以用GRU、LSTM或CNN等结构，但注意再次强调这种“预知未来”的特性仅仅在训练中才有可能，在预测阶段是不存在的，因此decoder在执行每一步时，不能提前使用后面步的输入。所以，如果用RNN结构，一般都只使用单向RNN；如果使用CNN或者纯Attention，那么需要把后面的部分给mask掉（对于卷积来说，就是在卷积核上乘上一个0/1矩阵，使得卷积只能读取当前位置及其“左边”的输入，对于Attention来说也类似，不过是对query的序列进行mask处理）。

> 敏感的读者可能会察觉到，这种训练方案是“局部”的，事实上不够端到端。比如当我们预测R时是假设Q已知的，即Q在前一步被成功预测，但这是不能直接得到保证的。一般前面某一步的预测出错，那么可能导致连锁反应，后面各步的训练和预测都没有意义了。
>
> 有学者考虑过这个问题，比如文章《Sequence-to-Sequence Learning as Beam-Search Optimization》把整个解码搜索过程也加入到训练过程，而且还是纯粹梯度下降的（不用强化学习），是非常值得借鉴的一种做法。不过局部训练的计算成本比较低，一般情况下我们都只是使用局部训练来训练seq2seq。

beam search

前面已经多次提到了解码过程，但还不完整。事实上，对于seq2seq来说，我们是在建模

$$p(\boldsymbol{Y}|\boldsymbol{X}) = p(Y_1|\boldsymbol{X})p(Y_2|\boldsymbol{X},Y_1)p(Y_3|\boldsymbol{X},Y_1,Y_2)p(Y_4|\boldsymbol{X},Y_1,Y_2,Y_3)p(Y_5|\boldsymbol{X},Y_1,Y_2,Y_3,Y_4) \tag{1}$$

显然在解码时，我们希望能找到最大概率的$\boldsymbol{Y}$，那要怎么做呢？

如果在第一步$p(Y_1|\boldsymbol{X})$时，直接选择最大概率的那个（我们期望是目标P），然后代入第二步$p(Y_2|\boldsymbol{X},Y_1)$，再次选择最大概率的$Y_2$，依此类推，每一步都选择当前最大概率的输出，那么就称为贪心搜索，是一种最低成本

的解码方案。但是要注意，这种方案得到的结果未必是最优的，假如第一步我们选择了概率不是最大的Y_1，代入第二步时也许会得到非常大的条件概率$p(Y_2|\boldsymbol{X},Y_1)$，从而两者的乘积会超过逐位取最大的算法。

然而，如果真的要枚举所有路径取最优，那计算量是大到难以接受的（这不是一个马尔可夫过程，动态规划也用不了）。因此，seq2seq使用了一种折中的方法：beam search。

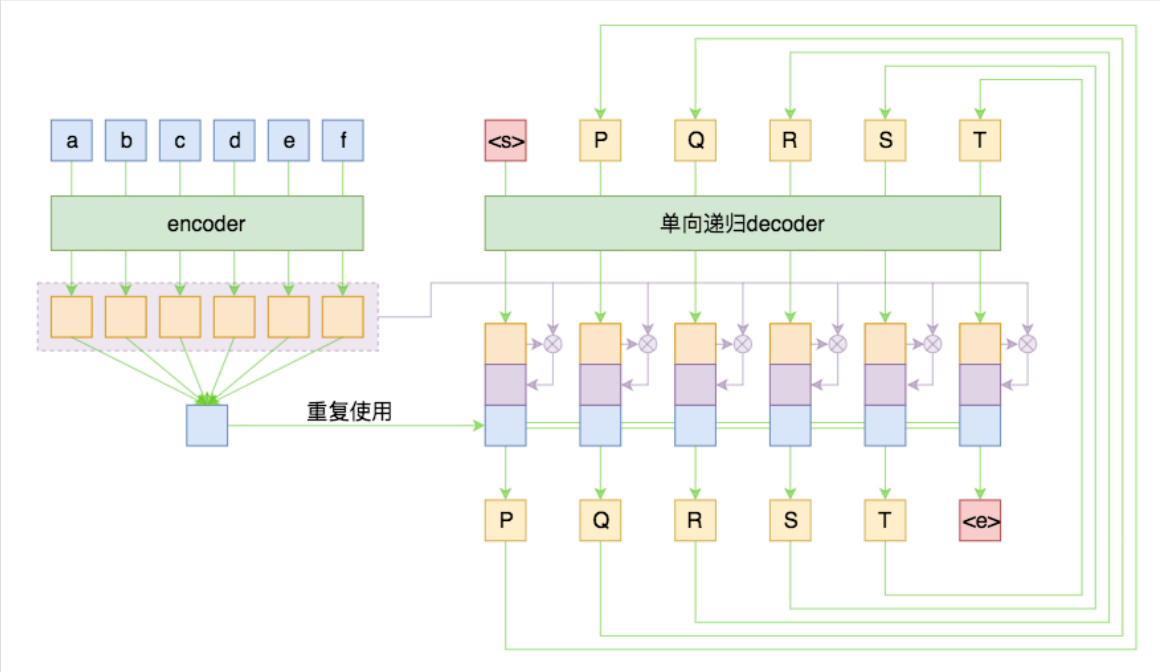
这种算法类似动态规划，但即使在能用动态规划的问题下，它还比动态规划要简单，它的思想是：在每步计算时，只保留当前最优的top_k个候选结果。比如取$top_k=3$，那么第一步时，我们只保留使得$p(Y_1|\boldsymbol{X})$最大的前3个Y_1，然后分别代入$p(Y_2|\boldsymbol{X},Y_1)$，然后各取前三个$Y_2$，这样一来我们就有$3^2=9$个组合了，这时我们计算每一种组合的总概率，然后还是只保留前三个，依次递归，直到出现了第一个<end>。显然，它本质上还属于贪心搜索的范畴，只不过贪心的过程中保留了更多的可能性，普通的贪心搜索相当于$top_k=1$。

seq2seq提升

前面所示的seq2seq模型是标准的，但它把整个输入编码为一个固定大小的向量，然后用这个向量解码，这意味着这个向量理论上能包含原来输入的所有信息，会对encoder和decoder有更高的要求，尤其在机器翻译等信息不变的任务上。因为这种模型相当于让我们“看了一遍中文后就直接写出对应的英文翻译”那样，要求有强大的记忆能力和解码能力，事实上普通人完全不必这样，我们还会反复翻看对比原文，这就导致了下面的两个技巧。

Attention

Attention目前基本上已经是seq2seq模型的“标配”模块了，它的思想就是：每一步解码时，不仅仅要结合encoder编码出来的固定大小的向量（通读全文），还要往回查阅原来的每一个字词（精读局部），两者配合来决定当前步的输出。



带Attention的seq2seq

至于Attention的具体做法，笔者之前已经撰文介绍过了，请参考《Attention is All You Need》浅读（简介+代码）。Attention一般分为乘性和加性两种，笔者介绍的是Google系统介绍的乘性的Attention，加性的Attention读者可以自行查阅，只要抓住query、key、value三个要素，Attention就都不难理解了。

先验知识

回到用seq2seq生成文章标题这个任务上，模型可以做些简化，并且可以引入一些先验知识。比如，由于输入语言和输出语言都是中文，因此encoder和decoder的Embedding层可以共享参数（也就是用同一套词向量）。这使得模型的参数量大幅度减少了。

此外，还有一个很有用的先验知识：标题中的大部分字词都在文章中出现过（注：仅仅是出现过，并不一定是连续出现，更不能说标题包含在文章中，不然就成为一个普通的序列标注问题了）。这样一来，我们可以用文章中的词集作为一个先验分布，加到解码过程的分类模型中，使得模型在解码输出时更倾向选用文章中已有的字词。

具体来说，在每一步预测时，我们得到总向量$\boldsymbol{x}$（如前面所述，它应该是decoder当前的隐层向量、encoder的编码向量、当前decoder与encoder的Attention编码三者的拼接），然后接入到全连接层，最终得到一个大小为$|V|$的向量$\boldsymbol{y}=(y_1,y_2,\dots,y_{|V|})$，其中$|V|$是词表的词数。$\boldsymbol{y}$经过softmax后，得到原本的概率

$$p_i=\frac{e^{y_i}}{\sum\limits_i e^{y_i}} \tag{2}$$

这就是原始的分类方案。引入先验分布的方案是，对于每篇文章，我们得到一个大小为$|V|$的0/1向量$\boldsymbol{\chi}=(\chi_1,\chi_2,\dots,\chi_{|V|})$，其中$\chi_i=1$意味着该词在文章中出现过，否则$\chi_i=0$。将这样的一个0/1向量经过一个缩放平移层得到：

$$\hat{\boldsymbol{y}}=\boldsymbol{s}\otimes\boldsymbol{\chi}+\boldsymbol{t}=(s_1\chi_1+t_1,s_2\chi_2+t_2,\dots,s_{|V|}\chi_{|V|}+t_{|V|}) \tag{3}$$

其中$\boldsymbol{s},\boldsymbol{t}$为训练参数，然后将这个向量与原来的$\boldsymbol{y}$取平均后才做softmax

$$\boldsymbol{y}\leftarrow\frac{\boldsymbol{y}+\hat{\boldsymbol{y}}}{2},\quad p_i=\frac{e^{y_i}}{\sum\limits_i e^{y_i}} \tag{4}$$

经实验，这个先验分布的引入，有助于加快收敛，生成更稳定的、质量更优的标题。

Keras参考

又到了快乐的开源时光~

基本实现

基于上面的描述，我收集了80多万篇新闻的语料，来试图训练一个自动标题的模型。简单起见，我选择了以字为基本单位，并且引入了4个额外标记，分别代表mask、unk、start、end。而encoder我使用了双层双向LSTM，decoder使用了双层单向LSTM。具体细节可以参考源码（Python 2.7 + Keras 2.2.4 + Tensorflow 1.8）：

https://github.com/bojone/seq2seq/blob/master/seq2seq.py

我以6.4万文章为一个epoch，训练了50个epoch（一个多小时）之后，基本就生成了看上去还行的标题：

> 文章内容：8月28日，网络爆料称，华住集团旗下连锁酒店用户数据疑似发生泄露。从卖家发布的内容看，数据包含华住旗下汉庭、禧玥、桔子、宜必思等10余个品牌酒店的住客信息。泄露的信息包括华住官网注册资料、酒店

入住登记的身份 信息及酒店开房记录，住客姓名、手机号、邮箱、身份证号、登录账号密码等。卖家对这个约5亿条数据打包出售。第三方安全平台威胁猎人对信息出售者提供的三万条数据进行验证，认为数据真实性非常高。当天下午，华住集团发 声明称，已在内部迅速开展核查，并第一时间报警。当晚，上海警方消息称，接到华住集团报案，警方已经介入调查。

生成标题：《酒店用户数据疑似发生泄露》

文章内容：新浪体育讯 北京时间10月16日,NBA中国赛广州站如约开打,火箭再次胜出,以95-85击败篮网。姚明渐入佳境,打了18分39秒,8投5中,拿下10分5个篮板,他还盖帽1次。火箭以两战皆胜的战绩圆满结束中国行。

生成标题：《直击:火箭两战皆胜火箭再胜 广州站姚明10分5板》

当然这只是两个比较好的例子，还有很多不好的例子，直接用到工程上肯定是不够的，还需要很多“黑科技”优化才行。

mask

在seq2seq中，做好mask是非常重要的，所谓mask，就是要遮掩掉不应该读取到的信息、或者是无用的信息，一般是用0/1向量来乘掉它。keras自带的mask机制十分不友好，有些层不支持mask，而普通的LSTM开启了mask后速度几乎下降了一半。所以现在我都是直接以0作为mask的标记，然后自己写个Lambda层进行转化的，这样速度基本无损，而且支持嵌入到任意层，具体可以参考上面的代码。

要注意我们以往一般是不区分mask和unk（未登录词）的，但如果采用我这种方案，还是把未登录词区分一下比较好，因为未登录词尽管我们不清楚具体含义，它还是一个真正的词，至少有占位作用，而mask是我们希望完全抹掉的信息。

解码端

代码中已经实现了beam search解码，读者可以自行测试不同的top_k对解码结果的影响。

这里要说的是，参考代码中对解码的实现是比较偷懒的，会使得解码速度大降。理论上来说，我们每次得到当前时刻的输出后，我们只需要传入到LSTM的下一步迭代中去，就可以得到下一时刻的输出，但这需要重写解码端的LSTM（也就是要区分训练阶段和测试阶段，两者共享权重），相对复杂，而且对初学者并不友好。所以我使用了一个非常粗暴的方案：每一步预测都重跑一次整个模型，这样一来代码量最少，但是越到后面越慢，原来是$\mathcal{O}(n)$的计算量变成了$\mathcal{O}(n^2)$。

最后的话

又用Keras跑通了一个例子，不错不错，坚定不移高举Keras旗帜~

自动标题任务的语料比较好找，而且在seq2seq任务中属于难度比较低的一个，适合大家练手，想要入坑的朋友赶紧上吧哈。

苏剑林. (Sep. 01, 2018). 《玩转Keras之seq2seq自动生成标题 》[Blog post]. Retrieved from https://spaces.ac.cn/archives/5861

@online{kexuefm-5861,
title={玩转Keras之seq2seq自动生成标题},
author={苏剑林},
year={2018},
month={Sep},
url={\url{https://spaces.ac.cn/archives/5861}},
}