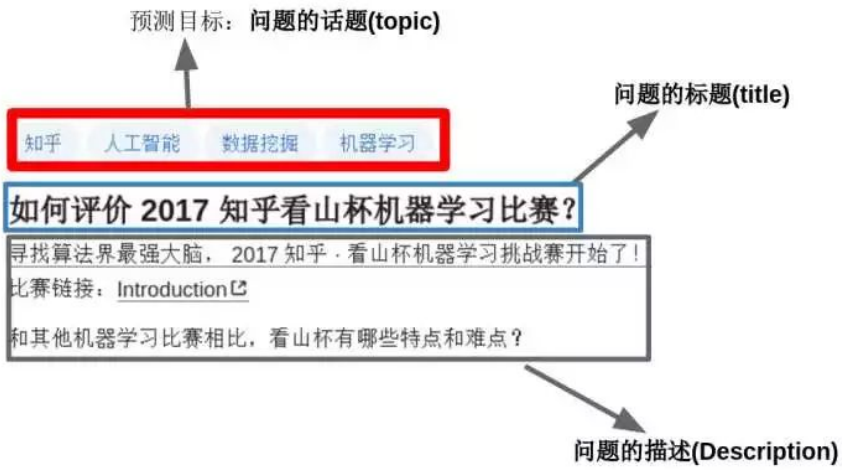
NLP大赛冠军总结：300万知乎多标签文本分类任务（附深度学习源码）

2017-11-24 达坂城大豆 阅 4356 转 20

七月，酷暑难耐，认识的几位同学参加知乎看山杯，均取得不错的排名。当时天池AI医疗大赛初赛结束，官方正在为复赛进行平台调试，复赛时间一拖再拖。看着几位同学在比赛中排名都还很不错，于是决定抽空试一试。结果一发不可收拾，又找了两个同学一起组队（队伍init）以至于整个暑假都投入到这个比赛之中，并最终以一定的优势夺得第一名。

1. 比赛介绍

这是一个文本多分类的问题：目标是“参赛者根据知乎给出的问题及话题标签的绑定关系的训练数据，训练出对未标注数据自动标注的模型”。通俗点讲就是：当用户在知乎上提问时，程序要能够根据问题的内容自动为其添加话题标签。一个问题可能对应着多个话题标签，如下图所示。



这是一个文本多分类，多label的分类问题（一个样本可能属于多个类别）。总共有300万条问题-话题对，超过2亿词，4亿字，共1999个类别。

1.1 数据介绍

参考 https://biendata.com/competition/zhihu/data/

https://biendata.com/competition/zhihu/rules/?next_url=%2Fcompetition%2Fzhihu%2Fdata%2F

总的来说就是：

数据经过脱敏处理，看到的不是“如何评价2017知乎看山杯机器学习比赛”，而是“w2w34 w234w54w909w2343w1'这种经过映射的词的形式，或者是”c13c44c4c5642c782c934c02c2309c42c13c234c97c8425c98c4c340'这种经过映射的字的形式。

因为词和字经过脱敏处理，所以无法使用第三方的词向量，官方特地提供了预训练好的词向量，即char_embedding.txt和word_embedding.txt，都是256 维。

主办方提供了1999个类别的描述和类别之间的父子关系（比如机器学习的父话题是人工智能，统计学和计算机科学），但这个知识没有用上。

达坂城大豆
关注 对话

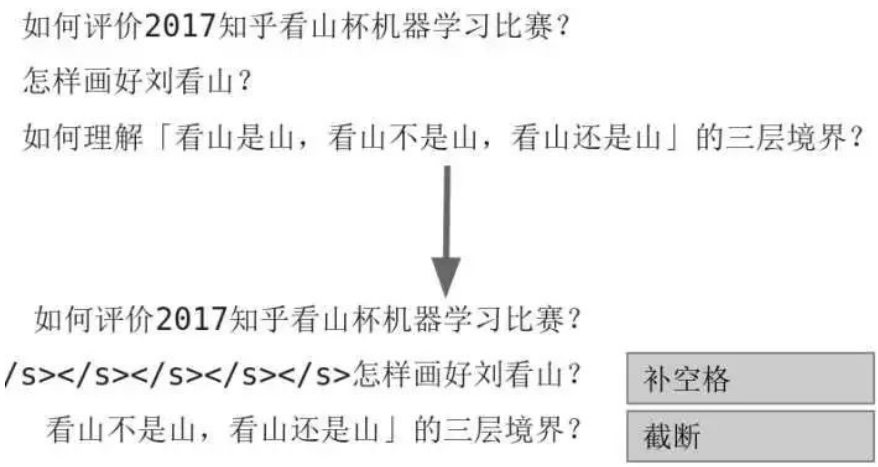
测试集包含21万条问题的标题（title），问题的描述(description)，需要给出最有可能的5个话题（topic）

1.2 数据处理

数据处理主要包括两部分：

char_embedding.txt 和 word_embedding.txt 转为numpy格式，这个很简单，直接使用word2vec的python工具即可

对于不同长度的问题文本，pad和截断成一样长度的（利用pad_sequence 函数，也可以自己写代码pad）。太短的就补空格，太长的就截断。操作图示如下：



1.3 数据增强

文本中数据增强不太常见，这里我们使用了shuffle和drop两种数据增强，前者打乱词顺序，后者随机的删除掉某些词。效果举例如图：

1.4 评价指标

每个预测样本，提供最有可能的五个话题标签，计算加权后的准确率和召回率，再计算F1值。注意准确率是加权累加的，意味着越靠前的正确预测对分数贡献越大，同时也意味着准确率可能高于1，但是F1值计算的时候分子没有乘以2，所以0.5是很难达到的。

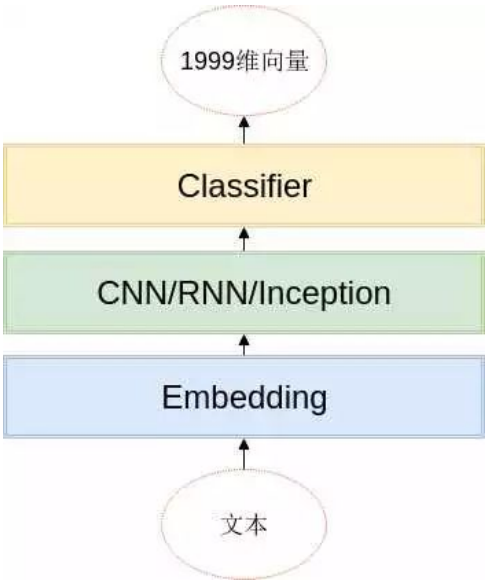
2 模型介绍

建议大家先阅读这篇文章，了解文本多分类问题几个常用模型：用深度学习（CNN RNN Attention）解决大规模文本分类问题

https://zhuanlan.zhihu.com/p/25928551

2.1 通用模型结构

文本分类的模型很多，这次比赛中用到的模型基本上都遵循以下的架构：



基本思路就是，词（或者字）经过embedding层之后，利用CNN/RNN等结构，提取局部信息、全局信息或上下文信息，利用分类器进行分类，分类器的是由两层全连接层组成的。

在开始介绍每个模型之前，这里先下几个结论：

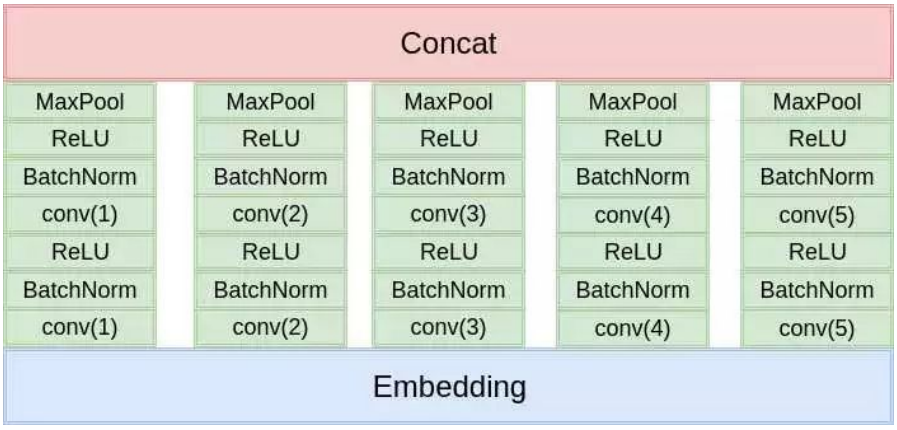
如果你的模型分数不够高，试着把模型变得更深更宽更复杂

当模型复杂到一定程度的时候，不同模型的分数差距很小

当模型复杂达到一定程度，继续变复杂难以继续提升模型的分数

2.2 TextCNN

这是最经典的文本分类模型，这里就不细说了，模型架构如下图：



和原始的论文的区别就在于：

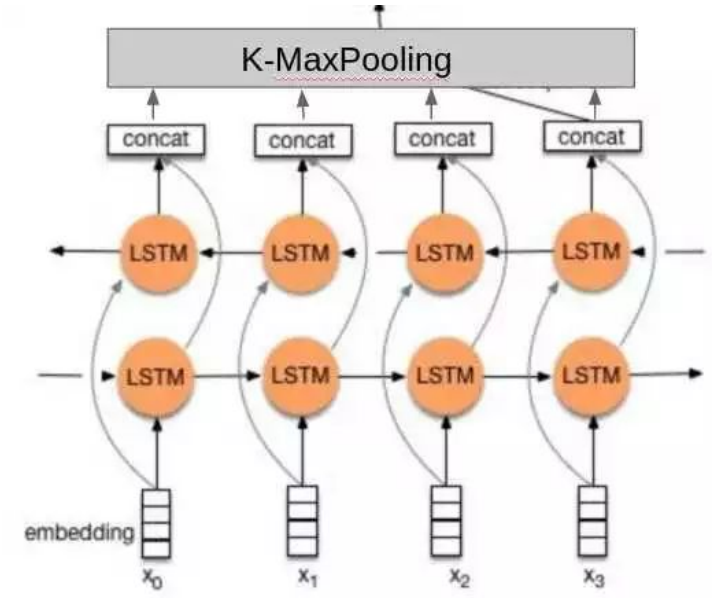
使用了BatchNorm

分类的时候使用了两层的全连接

总之就是更深，更复杂。不过卷积核的尺寸设计的不够合理，导致感受野差距过大。

2.3 TextRNN

没找到论文，我就凭感觉实现了一下：



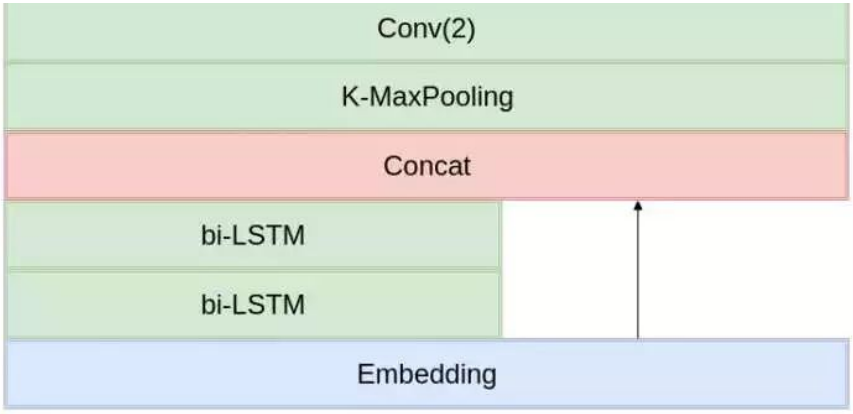
相比于其他人的做法，这里的不同点在于：

使用了两层的双向LSTM。

分类的时候不是只使用最后一个隐藏元的输出，而是把所有隐藏元的输出做K-MaxPooling再分类。

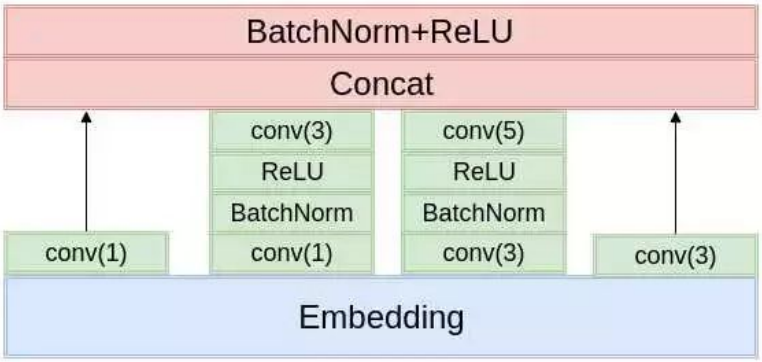
2.4 TextRCNN

参考原论文的实现，和RNN类似，也是两层双向LSTM，但是需要和Embedding层的输出Concat(类似于resnet的shortcut直连)。



2.5 TextInception

这个是我自己提出来的，参照TextCNN的思想（多尺度卷积核），模仿Inception的结构设计出来的，一层的Inception结构如下图所示，比赛中用了两层的Inception结构，最深有4层卷积，比TextCNN更深。



2.6 训练方法

要点：

基于词和基于字的模型要分开训，然后融合，一起训的效果不好

使用官方给的word-embedding.txt和char-embedding.txt初始化Embedding层的权重

刚开始训练的时候Embedding层的学习率为0，其它层的学习率为1e-3，采用Adam优化器（一开始的时候卷积层都是随机初始化的，反向传播得到的Embedding层的梯度受到卷积层的影响，相当于噪声）

训练1-2个epoch之后，Embedding层的学习率设为2e-4

每个epoch或者半个epoch统计一次在验证集的分数

如果分数上升，保存模型，并记下保存路径

如果分数下降，加载上一个模型的保存路径，并降低学习率为一半（重新初始化优化器，清空动量信息，而不是只修改学习率----使用PyTorch的话新建一个新优化器即可）

型。各个模型的分数都差不多，这里不再单独列出来了，只区分训练的模型的类型和数据增强与否。

类型	是否数据增强	分数
word	否	0.416-0.418
char	否	0.407-0.409
word	是	0.417-0.419
char	是	0.393-0.405

可以看出来

基于词的模型效果远远好于基于字的（说明中文分词很有必要）。

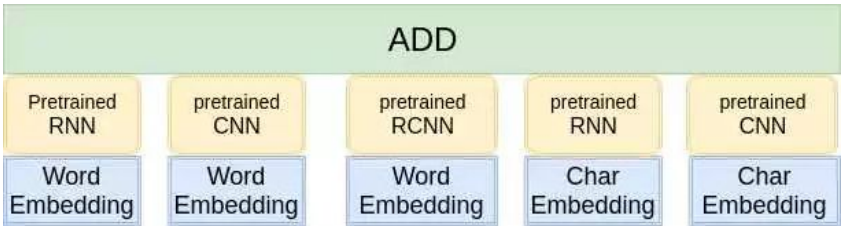
数据增强对基于词（word）的模型有一定的提升，但是对于基于字（char）的模型主要是起到副作用。

各个模型之间的分数差距不大。

2.8 模型融合

像这种模型比较简单，数据量相对比较小的比赛，模型融合是比赛获胜的关键。

在这里，我只使用到了最简单的模型融合方法-----概率等权重融合。对于每个样本，单模型会给出一个1999维的向量，代表着这个模型属于1999个话题的概率。融合的方式就是把每一个模型输出的向量直接相加，然后选择概率最大的5个话题提交。结构如图所示：



下面我们再来看看两个模型融合的分数：

模型1_分数	模型2_分数	分数	变量
RNN_0.4172	RCNN_0.4168	0.4240	模型（RNN与RCNN的结构差别）
RNN_0.4172	Inception_0.4162	0.4245	模型（RNN与Incetpion的结构差别）
RNN_0.4172	RNN_0.4189（数据增强）	0.4251	数据（是否进行数据增强），模型结构一模一样
RNN_0.4172	RNN_char_0.4084	0.4246	数据（word与char），模型结构一模一样

第一列的对比模型采用的是RNN（不采用数据增强，使用word作为训练数据），第二列是四个不同的模型（不同的结构，或者是不同的数据）。

我们可以得出以下几个结论：

从第一行和第二行的对比之中我们可以看出，模型差异越大提升越多（RNN和RCNN比较相似，因为他们底层都采用了双向LSTM提取特征），虽然RCNN的分数比Inception要高，Inception对模型融合的提升更大。

从第一行和第四行的对比之中我们可以看出，数据的差异越大，融合的提升越多，虽然基于字（char）训练的模型分数比较低，但是和基于词训练的模型进行融合，还是能有极大的提升。

采用数据增强，有助于提升数据的差异性，对模型融合的提升帮助也很大。

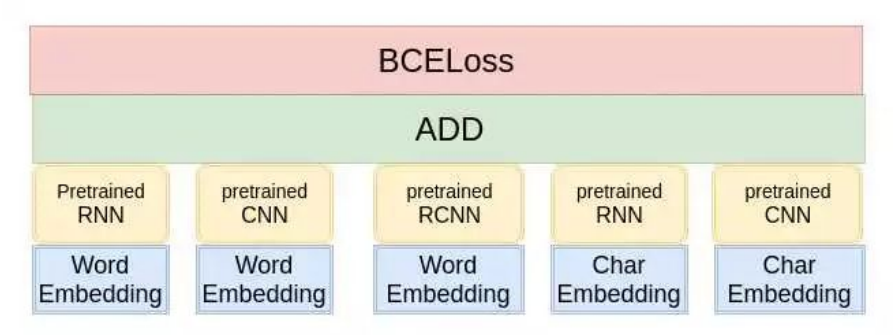
总结： 差异性越大，模型融合效果越好。没有差异性，创造条件也要制造差异性。

另外模型融合还有个规律：越往上越难提升,有些模型在你分数较低的时候，对融合提升很明显，当你分数较高的时候就没什么帮助，甚至会有干扰

2.9 MultiModel

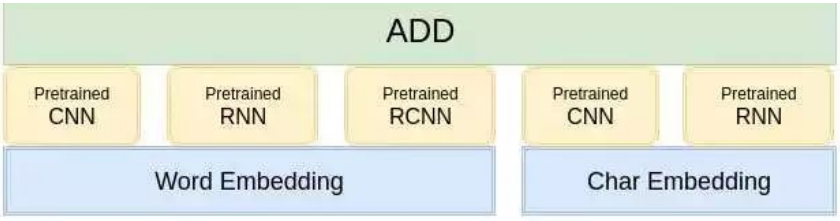
其实模型融合的方式，我们换一种角度考虑，其实就是一个很大的模型，每一个分支就像多通道的TextCNN一样。那么我们能不能训练一个超级大的模型？答案是可以的，但是效果往往很差。因为模型过于复杂，太难以训练。这里我尝试了两种改进的方法。

第一种方法，利用预训练好的单模型初始化复杂模型的某一部分参数，模型架构如图所示：



但是这种做法会带来一个问题： 模型过拟合很严重，难以学习到新的东西。因为单模型在训练集上的分数都接近0.5，已经逼近理论上的极限分数，这时候很难接着学习到新的内容。这里采取的应对策略是采用较高的初始学习率，强行把模型从过拟合点拉出来，使得模型在训练集上的分数迅速降低到0.4左右，然后再降低学习率，缓慢学习，提升模型的分数。

第二种做法是修改预训练模型的embedding矩阵为官方给的embedding权重。这样共享embedding的做法，能够一定程度上抑制模型过拟合，减少参数量。虽然CNN/RNN等模型的参数过拟合，但是由于相对应的embedding没有过拟合，所以模型一开始分数就会下降许多，然后再缓慢提升。这种做法更优。在最后提交模型复现成绩的时候，我只提交了七个这种模型，里面包含着不同子模型的组合，一般包含3-4个子模型。这种方式生成的权重文件也比较小（600M-700M左右），上传到网盘相对来说更方便。



2.10 失败的模型或没什么用的方法

MultiMode只是我诸多尝试的方法中比较成功的一个，其它方法大多以失败告终（或者效果不明显）

数据多折训练：因为过拟合严重，想着先拿一半数据训，允许它充分过拟合，然后再拿另外一半数据训。效果不如之前的模型。

Attention Stack，参考了这篇文章，其实本质上相当于调权重，但是效果有限，还麻烦，所以最后直接用等权重融合（权重全设为1）。

Stack，太费时费力，浪费了不少时间，也有可能是实现有误，提升有限，没有继续研究下去。

Boost，和第二名Koala的方法很像，先训一个模型，然后再训第二个模型和第一个模型的输出相加，但是固定第一个模型的参数。相当于不停的修正上一个模型误判的(可以尝试计算一下梯度，你会发现第一个模型已经判对的样本，即使第二个模型判别错了，第二个模型的梯度也不会很大，即第二个模型不会花费太多时间学习这个样本）。但是效果不好，原因：过拟合很严重，第一个模型在训练集上的分数直接就逼近0.5，导致第二个模型什么都没学到。Koala队伍最终就是凭借着这个Boost模型拿到了第二名，我过早放弃，没能在这个方法上有所突破十分遗憾。

TTA（测试时数据增强），相当于在测试的时候人为的制造差异性，对单模型的效果一般，对融合几乎没有帮助。

Hyperopt进行超参数查询，主要用来查询模型融合的权重，效果一般，最后就也没有使用了，就手动稍微调了一下。

label设权重，对于正样本给予更高的权重，训练模型，然后和正常权重的模型进行融合，在单模型上能够提升2-3个千分点（十分巨大），但是在最后的模型融合是效果很有限（0.0002），而且需要调整权重比较麻烦，遂舍弃。

用分类得到的词向量作为下一个模型的embedding的初始值，因为官方给的word embedding是用无监督的word2vec训练的，和有监督的分类问题还是有一定偏差的。没有深入研究下去，对单模型应该是有提升，但是对融合可能没什么帮助。

3 结束语

我之前虽然学过CS224D的课程，也做了前两次的作业，但是除此之外几乎从来没写过自然语言处理相关的代码，能拿第一离不开队友的支持，和同学们不断的激励。

这次比赛入门对我帮助最大的两篇文章是用深度学习（CNN RNN Attention）解决大规模文本分类问题

https://zhuanlan.zhihu.com/p/25928551

和deep-learning-nlp-best-practices

http://ruder.io/deep-learning-nlp-best-practices/index.html

第二届是国外某博士写的，当时我已经把分数刷到前二，在家看到了这篇文章，收为观止，解释了我很多的疑惑，提到的很多经验总结和我的情况也确实相符。https://zhuanlan.zhihu.com/p/28923961

转藏到我的图书馆 献花（0） 分享： 微信

来自： 达坂城大豆 > 《Python》 以文找文 | 举报

下一篇：在GUI窗口中绘制一个茅台股票K线图

猜你喜欢

类似文章 更多

- 当深度学习遇见自动文本摘要
- 用 RNN 训练语言模型生成文本
- AI技术讲座精选：NLP 模型到底选 RNN 还...
- 使用 TensorFlow 做文本情感分析
- 鹏元数据：自然语言处理——使用Word2Ve...
- 每天接触大量论文，看看他们是怎样写笔记...
- 深度学习史上最全总结（文末有福利）
- WOT2016黄伟：基于深度学习的情感分析

精选文章

- 读书简介：你一年的8760小时
- 家里娃娃两三个，成天打闹不已，家长该怎么...
- 那张名为英语四六级的彩票
- 超清电影《八零后的那些事》
- 千万美女图片尽在手中[上]
- 给所有今天心情不好的人。。。
- 要想成功,必须做到这些
- 你总是讨好他人吗

发表评论

请 **登录** 或者 **注册** 后再进行评论 社交帐号登录：