文本分类有哪些论文中很少提及却对性能有重要影响的tricks?

原创 夕小瑶 夕小瑶的卖萌屋 2019-01-21

来自专辑
卖萌屋@自然语言处理

前言

一年前小夕在知乎上提问过这么一个问题

> 文本分类有哪些论文中很少提及却对性能有重要影响的tricks?
> 链接：https://www.zhihu.com/question/265357659/answer/578944550

当时正好在刷一个比较有趣的task，结果发现奇奇怪怪的tricks可以带来不少的性能收益。再加上后来为了验证一个小idea跑了一堆公开的文本分类数据集，虽然idea没有多亮，倒是积累和摸索了不少刷性能的tricks┐(￣▽￣")┌然后呢，小夕后续又用这些tricks刷了不少相关的比赛（哪怕是文本匹配这种特殊的文本分类问题），发现baseline+一堆tricks+简单集成就可以随随便便刷到一个文本分类的水比赛的top10甚至top3，甚感调参和tricks的重要性。

然鹅，最近好一段时间都没有文本分类这个基础问题了，感觉都快忘了，趁着还有点模糊的记忆就整理下来分享给大家叭～希望能在大家刷论文实验、比赛或实际项目的时候提供点帮助或者启发。

首先来一个结论，tricks用的好，调参调的妙，TextCNN也能吊打绝大多数花里胡哨的深度模型。tricks没用好，SOTA模型也会性能差的让你怀疑人生。下面就不分重点，没有逻辑的开始本文辣。

关于分词器

中文也好，英文也好，拿过来数据集无可避免的就是要看看要不要做分词（有的小伙伴以为英文数据集就完全不用分词真的让人很无奈鸭），如果要做，就要纠结分词器的选择了。

> 路人丙：我厂有全方位吊打各种开源分词工具的分词器了
> 小夕：好了你可以往下划了

首先就有一个问题，**真的是算法越"先进"的分词器就会给下游任务带来越好的性能吗?**

很多人走到这一步的时候会忽略一个东西，**词向量！！！**

其实比起分词算法本身的先进程度，在神经网络使用预训练词向量的大背景下，**确保分词器与词向量表中的token粒度match其实是更更重要的事情！**毕竟哪怕你词分的再好，一旦词向量表里没有的话，那么就变成OOV了，分的再好也木用了┐(￣▽￣")┌（除非你不嫌麻烦多写点代码去对相对于词向量表的OOV进行特殊处理，反正我一般嫌麻烦┐(ㄏ▽ㄣ)┌）于是这里就有了两种情况。

1. 已知预训练词向量的分词器

一般像word2vec、glove、fasttext这些官方release的预训练词向量都会公布相应训练语料的信息，包括预处理策略如分词等，这种情况真是再好不过了，不用纠结，如果你决定了使用某一份词向量，那么直接使用训练该词向量所使用的分词器叭！此分词器在下游任务的表现十之八九会比其他花里胡哨的分词器好用。

2. 不知道预训练词向量的分词器

这时就需要去"猜"一下分词器了。怎么猜呢？首先，拿到预训练词向量表后，去里面search一些特定词汇比如一些网站、邮箱、成语、人名等，英文里还有 `n't` 等，看看训练词向量使用的分词器是把它们分成什么粒度，然后跑几个分词器，看看哪个分词器的粒度跟他最接近就用哪个，如果不放心，就放到下游任务里跑跑看啦。

当然，最理想的情况当然是先确定最适合当前任务数据集的分词器，再使用同分词器产出的预训练词向量啦。可惜互联网上不可能有那么多版本的公开词向量供选择，因此自己在下游任务训练集或者大量同分布无监督语料上训练词向量显然更有利于进一步压榨模型的性能。不过，怎么为当前的任务去预训练一份儿好用的词向量又够写一篇文章的。。这里就不展开讲啦，小夕以后再写～（没关注小夕的赶紧关注！）

当然，除了分词器跟词向量表要match上，另外还要保证大小写、OOV的定义等跟词向量表match上。如果使用了一个区分了大小写的词向量表，但是你还将下游任务的单词全都小写，那么不用想了，绝对性能丢N多个百分点。

关于中文字向量

> 路人丁：好麻烦，我不分词了，我要用字向量了哼
> 小夕：别逃(￣▽￣)

如果你真的将char-level作为主力，那么别忘了中文的字向量也要预训练！并且预训练的时候记得把**窗口开大一些**，不要直接使用word-level的窗口大小哦，其他预训练超参数也随手调一调更好了，绝对比随机初始化的字向量明显的好。

如果数据集噪声很严重

这里噪声严重有两种情况。对于数据集D(X, Y)，一种是X内部噪声很大（比如文本为口语化表述或由广大互联网用户生成），一种是Y的噪声很大（一些样本被明显的错误标注，一些样本人也很难定义是属于哪一类，甚至具备类别二义性）。

对于前一种噪声，一个很自然的想法是去使用语言模型或者基于编辑距离去做文本纠错，然鹅实际中由于专有名词和超出想象的“假噪声”存在，在实际场景中往往效果并不是很好。

这里小夕一般有两种思路，一种是直接将模型的输入变成char-level（中文中就是字的粒度），然后train from scratch（不使用预训练词向量）去跟word-level的对比一下，如果char-level的明显的效果好，那么短时间之内就直接基于char-level去做模型叭～

如果性能差不太多，或者char的已经做到头了，想做一下word-level呢？

不要急，先帮小夕买根棒棒糖呗(￣▽￣)

一个很work但是貌似没有太多人发现的trick就是使用**特殊超参的FastText**去训练一份词向量啦。

为什么说特殊呢?一般来说fasttext在英文中的char ngram的窗口大小一般取值3～6，但是在处理中文时，如果我们的目的是为了去除输入中的噪声，那么我们可以把这个窗口限制为1～2，这种小窗口有利于模型去捕获错别字（想象一下，我们打一个错误词的时候，一般都是将其中的一个字达成同音异形的另一个字），比如word2vec学出来的“似乎”的最近词可能是“好像”，然而小ngram窗口fasttext学出来的“似乎”最近词则很有可能是“是乎”等内部包含错别字的词，这样就一下子让不太过分的错别字构成的词们又重新回到了一起，甚至可以一定程度上对抗分词器产生的噪声（把一个词切分成多个字）。当然，如果数据集很干净的话，这样训练词向量的话可能就gg了。

而对于后一种噪声的情况（即Y中的噪声），一种很直接的想法是做标签平滑，然而小夕在实战中使用多次发现效果并不是太明显。

最后总结的trick是，首先忽略这个噪声，强行的把模型尽可能好的训出来，然后让训练好的模型去跑训练集和开发集，取出训练集中的错误样本和开发集中那些以很高的置信度做出错误决策的样本（比如以99%的把握把一个标签为0的样本预测为1），然后去做这些bad cases的分析，如果发现错误标注有很强的规律性，则直接撸一个脚本批量纠正一下（只要确保纠正后的标注正确率比纠正前明显高就行）。

如果没有什么规律，但是发现模型高置信度做错的这些样本大部分都是标注错误的话，就直接把这些样本都删掉吧～常常也可以换来性能的小幅提升，毕竟测试集都是人工标注的，困难样本和错标样本不会太多。

baseline选用CNN还是RNN？路线沿着CNN还是RNN走？

在文本分类中真的不要太纠结这个问题，个人倾向于CNN，主要是因为跑得快呀。。。可以多跑几组实验，多好。而且实

际经验感觉TextCNN这种基础款CNN模型不仅实现特别容易，而且很容易成为一个数据集上的很强的baseline（除非这个分类任务很难），花一两个小时把这个baseline做出来后再去做其他模型一点也不迟～也有助于早期就能纠正大方向。

而如果要谈到客观的思路决策上，那就去花一个小时好好看一下数据集吧～如果你感觉数据集里很多很强的ngram可以直接帮助生成正确决策，那就CNN起步吧。如果感觉很多case都是那种需要把一个句子看完甚至看两三遍才容易得出正确tag，那就RNN起步吧。

当然，如果数据大，又有显卡，还可以尝试Transformer。时间多的话，还可以CNN、RNN的模型都跑出来简单集成一下。

Dropout加在哪里

word　embedding层后、pooling层后、FC层　（全联接层）后，哦了。起步阶段dropout概率保持统一，有时间再单独微调就好（从来没有这个时间过）。

至于偶尔有人吹捧的word　　dropout策略（将一些token随机mask成[PAD]，或者说0。注意这个操作跟dropout加在embedding层后不等价哈），最后有时间的话试一下就好，亲测在dropout调好的情况下一般并不会发挥多大作用。

关于二分类

二分类问题一定要用sigmoid作为输出层的激活函数？当然不是，尝试一下包含俩类别的softmax吧。可能多一条分支就多一点信息叭，虽然后者在数学形式上更丑一点，但是实践中常常带来零点几个点的提升也是比较玄学了。

关于多标签分类

如果一个样本同时拥有多个标签，甚至标签同时还构成了DAG（有向无环图），不要着急，先用binary-cross-entropy训出个baseline来（即把每个类别变成一个二分类问题，这样N个类别的多标签分类问题就变成了N个二分类问题），毕竟这个都在tensorflow里有现成API了，即tf.nn.sigmoid_cross_entropy_with_logits。因此实现代价很小。

然后你还可能惊喜的发现，这个baseline做好后好像多标签问题不大了，DAG问题自己也基本解决了（虽然模型层并没有专门针对这个问题作处理），然后就可以安心做模型辣。什么？问题木有解决？去查论文吧╮(￣▽￣"")╭小夕还没有接触过这方面太难的数据集。

类别不均衡怎么办

像网上说的那样赶紧各种上采样下采样boosting策略用起来？nono，正负样本比才9:1的话，继续做你的深度模型调你的超参吧，模型做好后你会发现这点不均衡对模型来说不值一提，决策阈值也完全不用手调。但！是！如果你发现经常一个batch中完全就是同一个类别的样本，或者一些类别的样本经过好多batch都难遇到一个的话，均衡就非常非常有必要了。类别不均衡问题传送门->

【小夕精选】如何优雅而时髦的解决不均衡分类问题

别太纠结系列

1. 别太纠结文本截断长度使用120还是150
2. 别太纠结对性能不敏感的超参数带来的开发集性能的微小提升
3. 别太纠结未登陆词的embedding是初始化成全0还是随机初始化，别跟PAD共享embedding就行
4. 别太纠结优化器用Adam还是MomentumSGD，如果跟SGD的感情还不深，就无脑Adam，最后再用MomentumSGD跑几遍

还是不会用tricks但是就是想跑出个好结果怎么办

BERT了解一下。

Over。

暂时想起来的就是这些啦，剩下有想起来的tricks小夕会更新到知乎上，传送门：

> https://www.zhihu.com/question/265357659/answer/578944550

话说，小夕跟大家分享了这么多tricks，亲爱的们有没有秘藏tricks在评论区分享给小夕呢(￣▽￣)
