ML&DEV[13] | bad case分析

原创 机智的叉烧 CS的陋室 2020-02-23 21:00

Because of You

Josh Vietti - Best of Both Worlds

【ML&DEV】

这是大家没有看过的船新栏目！ML表示机器学习，DEV表示开发，本专栏旨在为大家分享作为算法工程师的工作，机器学习生态下的有关模型方法和技术，从数据生产到模型部署维护监控全流程，预备知识、理论、技术、经验等都会涉及，近期内容以入门线路为主，敬请期待！

往期回顾：

- ML&DEV[8] | 算法在岗一年的经验总结
- ML&DEV[9] | gRPC初体验
- ML&DEV[10] | gRPC的应用
- ML&DEV[11] | 浅谈模型的局限性
- ML&DEV[12] | ML中的数学学习

模型建立，问题解决总不一帆风顺，在面临一些结果不佳的问题的时候，不应该是想着换模型之类的，而应该有更加丰富的一些方法，而为了有针对性的解决问题，其实非常有必要对bad case进行深入分析，本文就来给大家谈谈怎么去做这种bad case分析。

什么是bad case

应该是业内的专业名词吧，简单解释一下。bad是坏，case是案例，说白了就是模型或者具体规则无法很好预测的那些案例。以分类模型为例，本来一个样本应该是负类的，结果预测为正类，那这个样本就能被称为bad case。

为什么要做bad case分析

个人理解bad case应该是日常提升模型效果的关键手段。主要因为通过bad case分析，你能知道你当前方案的主要缺陷在什么位置，方便定位问题，这就和考完试订正错题一样，分析自己错在哪，应该怎么修正一样。一般的，bad case能够检查出下面这些问题，或者说能有如下作用：

- 掌握你的规则、特征或者策略的覆盖情况，例如有没有宁可错杀一千也不放过一个之类的错误，特征异常值等。
- 你的策略、特征工程等方法是否生效，bad case分析是验证这个问题的唯一方法。
- 特征的表征方式是否正确，例如词向量，词权重等，例如大领域训练的词向量拿到小领域可能会出现这种情况，或者是某些特征需要做转化或者处理。
- 有时候你会发现模型其实预测是对的，但是标注是错的。这时候你估计要回头去看看你的数据质量了。
- 某些错误的严重程度，可以通过出现的频次和比例进行对比得到，严重的、占比大的才是主要矛盾。

如何做bad case分析

如何做bad case分析其实是一个非常考验经验的，就像高中同样订正答案分析问题不同的人分析的结论可能不同，最终达到的效果也是不同的，因此掌握好的bad case分析，是非常重要的。

例如我们有一套方案，这里涉及一个模型加一些上下游的策略，发现效果并不好。

第一步，肯定是抽取bad case，抽取的规则很简单，预测值和标注值不同的都给拿出来。要是分析的话，为了具有统计学意义，一般是100个，别以为很多，其实很快的。

第二步，开始一个一个的分析问题，具体分析的点如下：

- 看看标注和预测那个是真正正确的，是存在标注错误的可能性的，如果是标准数据集，可以跳过。如果这种问题太多，请重新评估一下标注结果。

- 追溯这个case计算的整个流程，从开头特征分解，规则，词向量读取，模型计算（模型计算可以先当做黑盒）等，举几个例子吧，未登录词问题，分词问题，词典质量或者覆盖度不足，模型效果不佳等，查看里面那个步骤的输出结果不符合预期。（大系统内的话，就打DEBUG日志，如果是小程序，那就print即可）。
- 定位问题以后开始查看问题出现的原因，原因可能很深，但这个找原因的目标在于找可能的解决方案。要是暂时找不到不要紧，一般是在你看了十来个后你去总结可能会更好，毕竟你最终要处理的不是一个case，而是一整个问题。

第三步，总结各种问题的共性和出现频次，总结特定部分问题的解决方案，给每一种问题的难度和影响面排序，说白了就是优先解决主要矛盾，我们需要分析出主要矛盾在哪里。

第四步，制定解决方案，评估解决方案的成本与预期能够提升的量，一般而言最终结果的召准是有一条及格线的，可以根据提升量预估和及格线之间的差距来确定自己有没有必要进行更多的提升策略。

按着这个流程去分析，一般都会分析到你当前需要采取的策略和问题，剩下的，我更愿意称为经验，你不知道自己不知道的东西，是怎么都想不出来的，所以才说实践出真知，出经验。

后记

这种分析其实经历了好几次，几乎每个项目都需要有这么一个过程，也发现一些比较有意思的结论。

- 多次case分析其实可以发现，大部分问题往往都不在模型上，而是在整个处理链路上出现了问题。
- 模型问题，一般是难解的，因为模型难以调控，甚至我们没有把握认为新的模型就一定能解决这个问题。
- 在科研上，你要解决特定的bad case所提出的方法其实就是创新点了，RNN为了处理序列关系而产生，LSTM在RNN基础上为了解决梯度消失等问题而产生，这不就是通过bad case分析得到的结论吗。
- 先考虑一个很多简单的方案，快速实现构建DEMO，然后分析改进提升，是工业界一个非常常规的处理方案，在科研界叫做"先重现前人的成果"，其实都是一回事，先把流程弄出来，然后再来改进，就和换零件一样。
- 集中精力，节约成本地解决主要矛盾。例如分词问题很多场景都会存在，但是其实通过bad case分析往往会发现其实这都不是主要问题，如果并不是目前的主要问题，其实不用动，且分词功能依赖会很多，修改起来下游变动很大，所以一般修改分词模型不会是主要的改进模型方案。
- 道理都懂，但是做不做是另一回事。这个道理希望大家都要做起来，而不仅仅是懂，偷懒都能理解，但是这个分析的懒你偷了，花的时间可能会更多。

我是叉烧，欢迎关注我！

叉烧，OPPO搜索算法工程师。19届北京科技大学数理学院统计学硕士（保研），17届北京科技大学信息与计算科学、金融工程本科双学位。论文7篇，1项国家自科参与人，国家级及以上会议4次，1次优秀论文，国家奖学金，北京市优秀毕业生。曾任去哪儿网大住宿事业部产品数据，美团点评出行事业部算法工程师。

微信 zgr950123
邮箱 chashaozgr@163.com
知乎 机智的叉烧

微信个人公众号
CS的陋室