2019 BDC 比赛总结

2019中国高校计算机大赛已正式结束。对于我个人而言，这是第一次进入算法比赛的TOP10。我们队伍在初赛排在大概50名左右，复赛B榜8名，最终决赛第9名，这里是自己比赛的最终提交方案，求star。。。。。。。。。。本次总结是争对决赛TOP5方案的，毕竟自己做的跟他们比还有一些差距，我会尽自己所能的将他们做的全部方案融合进这一篇blog中，以此来帮助到我和大家。

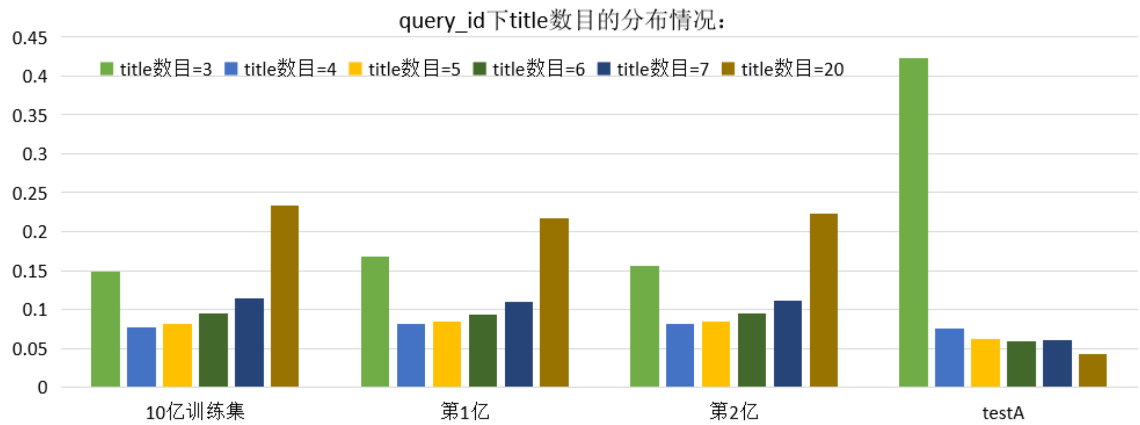
blog中直接引用了很多TOP5方案的PPT，如果有哪些组不愿意将自己的PPT或者方案公开，请联系我邮箱997647204@qq.com，我将第一时间在blog中删除。另外，相关nlp比赛想组队也可以联系我，求大佬carry。

总的来说，这次比赛前排TOP10的方案中几乎都包括了lgb和esim两个模型。而且在esim模型中除了第五名的一组，其他的都加入了手工特征。我记得评委在现场问过一句话--"我看过前面的很多方案，他们做的跟你们大体一样，但为啥各自的线上qAuc差别很大"，我已经记不得当时的大佬是怎么回答，前排的成绩相对来说还说还比较接近，各自的差异可能仅仅是由于一处到两处的做法不同，这一点在特征工程还不太明显，但是在后面nn结构设计尤为明显，TOP3都争对esim做了一些很小的改动，废话不多说了，下面直接进入到赛题分析部分。

赛题分析

搜索中一个重要的任务是根据query和title预测query下doc点击率，本次大赛参赛队伍需要根据脱敏后的数据预测指定doc的点击率。

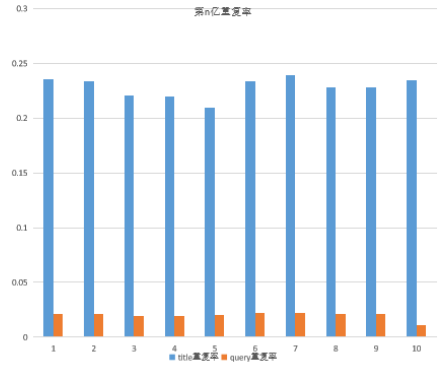
训练集为10亿，A榜测试集为2KW， B榜测试集为1亿。



第二名的队伍分析了queryid下title数目的分布情况，大家看一下这个柱状图，我们只列出来了title数目为3、4、5、6、7和20的分布情况，训练集中queryid下title数量为20的比较多，A榜测试集中queryid下title数量为3的比较多，我们通过线下测试发现，queryid下title数目为3的qauc值远低于title数目为20的qauc值，这也说明了为什么线上线下得分差异比较大，而B榜测试集的queryid下的title数量分布情况和训练差不多，因此B榜得分相对与A榜得分高了很多。

上面我原封不动的摘抄了他们的PPT，他们的大致意思就是相对于testA，testB query_id下title数目的分布情况更接近于训练集，但是他们这里并未给出testB的分布情况，可能是由于时间情况吧。假设testB和训练集的分布情况差异也很大的话，那么我们可能需要对训练集进行一些采样来拟合testB构建验证集进行本地验证，甚至可能需要拟合训练集分布来进行训练。

testA 2kw数据与train重复的query占比39.25%，title有71.18%，query-title对: 29.189%



1）对于重复title：文本信息+历史信息

2）对于新的title：文本和搜索领域相关特征比较重要

上图也来自第二名队伍的数据分析。其实这方面工作我们也做了，主要就是看看query和title对在测试集的重复率，当时我们是根据这重复率来对训练集进行采样拟合测试集A，我们考虑的是如果对于重复的title和query，nn预测出的点击率可能会比较高（因为它以前看过这个），而对于新的title和query，nn可能预测出的点击率可能会比较低（它以前没看过这个），因此如果测试集中大量都是训练集中没出现的query和title，那么整体预测出的点击率分布会偏低，qAuc这一排序指标就不太准，可惜的是当时我们做的这份工作反而把整体模型的性能拉低了。我猜测可能是由于训练集和测试集的时间戳是连续的，直接进行随机采样可能会打乱这时间戳的连续性，决赛中TOP10方案好像没有随机采样训练集的。（当然也存在另外一种可能，那就是我的代码有BUG，如果你有拟合测试集的成功案例的话希望能在讨论区跟我反馈一下）。

特征工程

特征工程大家做的差异很大，大致可以分为下列这几类

1、长度特征

- query的长度和title的长度
- query和title公共词的数量以及占比
- query和title长度的差值和比值
- query组内的title长度均值
- ...其实跟这方面相关的特征还有很多，比如可以对query和title去重之后再进行一次操作，但我认为这一个特征不是各自模型之间最大的差异。

2、 频次特征（强特）

- nunique特征，也可以理解为query和title出现的频次，不同query下title的数目，不同title下query的数目（对query id进行groupby之后再对title进行unique）

  下面是第四名对上述特征的解释：query次数表示用户看到不同title的个数。个人理解看的越少说明有足够满意的title。（包含用户行为信息）title的频率包含了title的热度，频率越高说明该title的热度越高。

3、匹配特征

- 最长公共子序列，最长公共子序列在query中的跨度，最长公共子序列在title中的跨度，最长子序列长度/title长度等特征
- 前缀后缀特征
- Difflib(计算query和title之间的diff ratio，marchs表示共现词长度和)， sequencematcher

$$2 * matchs / [len(q) + len(a)]$$

- LongShareRate，最长共现语句 s，计算

$$\begin{cases} \dfrac{len(s)}{\min(len(q), len(a)) + 1} \\ \dfrac{len(s)}{len(a) + 1} \end{cases}$$

- fuzzywuzzy模糊匹配
- N-gram特征
- 编辑距离（Levenshein)
- jaccard距离

4、熵特征

- query、title的信息熵

5、 图特征

- pagerank，分别以query和title为节点构建有向图，计算query和title的文本重要性。

$$R_{t+1} = dMR_t + \frac{1-d}{n}1$$

6、 相似度特征

下面一些相似度特征可以只选取query的前m个词，以及title的前n个词，因为对于ctr问题，前面词的重要性更加高。

首先定义相似度函数：有欧氏距离、曼哈顿距离、Canberra距离、汉明距离、余弦距离等等，具体可在scipy的spatial.distance下搜寻。

下面是第四名队伍给出的距离函数重要性排序：

将词向量进行句向量化，计算句向量之间的距离。（数字表示lgb特征重要性）

Cosine (2)

Cityblock (5)

Canberra (4)

Euclidean (1)

Minkowski (3)

BrayCurtis (4)

之后就要选择争对哪些向量了，可以直接得到的有tfidfvectorizer和countvectorizer出来的稀疏向量，对于这类向量的相似度的计算可以尝试skelarn.pairwise下的那些函数（经测试，它们的计算速度较快）

而对于word2vec和fasttext出来的向量组，有下列三种方式转成句向量：

- 直接等权平均
- 根据tf_idf加权，
- Smoothed Inverse Frequency Embedding，第二名队伍分析中指出这一种方法会比tf_IDF加权上升2个千

  word2vec和fasttext转成句向量后就可以直接通过相似度函数计算相似度特征

除去上面这些相似度特征，还有下面一些函数库计算的相似度特征

- bm25

7、 点击率特征（目标编码特征）

自己也尝试过去构建点击率特征，在初赛的时候总体性能还行，但是到了复赛却崩了，这是一个相当容易导致模型过拟合的特征，使用的时候技巧性比较强，各组争对过拟合也有自己的方案。

首先说第五名的方案：他们组使用使用前9亿5000万数据进行统计title的ctr，使用后5000万做训练，最终出

现了严重的过拟合。解决策略是将训练两个lgb，第一个包括ctr但不包括title相关特征，第二个包括title相关特征，两个lgb的特征是完全没有重合的地方的。由于lgb能够自动组合特征，将ctr和title两个特征拉开，可以防止lgb记住ctr和title的组合（也就是防止lgb将ctr和title关联到一起），以此来遏制过拟合。同时这两个lgb差异性比较大，这样他们的最终融合收益也比较高。

第四名的方案：

三、点击率特征：

CTR=点击量/出现次数（针对Title）

1、主要解决冷启动的文本（ctr=0.25）

2、频率低的文本的点击率（click/(sum+1)）

统计后 9 亿训练集 title 出现的次数（num）和点击的次数（click），将其 merge到训练集数据集，处理冷启动 title：首先使用 pandas 对 num 列进行 fillna（3）,click列 fillna（1），然后 num 列 +1，最后 click/num 获得点击率特征。

第二名的方案：

1、CTR特征

使用转化率特征的时候如何避免信息泄露?

对于CTR，使用了贝叶斯平滑

$$\hat{r} = \frac{C + \alpha}{I + \alpha + \beta}$$

CTR使用方式调研

方案	分数
5折不抽样	0.6012
5折0.5抽样	0.5958
10折不抽样	0.6014

第三名的方案（这也是我觉得解决的最好的一个点击率特征方案）：

根据关键词提取点击率特征：

1. 根据 TFIDF 对 query 提取一个关键词，title 提取两个关键词（拼接成字符串）
2. 将得到的关键词字符串hash映射到int型，降低提取时内存占用并提高速度
3. 然后提取query点击率、title点击率以及联合点击率

避免数据泄露：

- 5折交叉提取
- 提取训练集特征之前随机丢弃一半数据
- 贝叶斯平滑

他们组跟前面组最大的差别就是对query和title提取关键词，这样可以大幅减少空值。

8、 其余特征

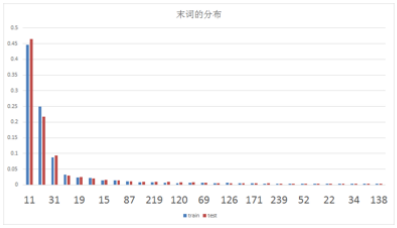
- 词位置特征（query前10个词在title中的位置）
- Proximity
- title质量分数特征

假设前提	1）一个query中用户点击了 i 篇title和另一个query中用户点击了 j（j！=i）篇title，二者被点击的title对于query的重要程度不一样。 2）一个query中用户有 i 篇title未点击和另一个query中用户有 j（j！=i）篇title未点击，二者未点击的title对于query的重要程度也不一样。
计算方式	1）对于点击的title，该title质量得分等于 1 - 该query下的title点击率 2）对于未点击的title，该title质量得分等于0 - 该query下的title点击率 3）title的最终得分是包含该title的所有query下该title的得分总和
计算公式	$\text{Score}_{title} = \sum_{\text{包含}title\text{的}query} lable_{query} - \text{点击率}_{query}$

效果情况	初赛qAUC	复赛qAUC
title 质量分数特征的值作为预测结果	0.570615	0.572866

- 末词（tag）的信息数据挖掘

列名	类型	示例
query_id	int，一个query的唯一标识	1
query	字符string，term空格分割	"学节跳动"
title	字符string，term空格分割	"学节跳动 百科"
label	int，取值{0, 1}，有点击为1，无点击为0	1



（四）末词（tag）的信息数据挖掘

特征	特征解释
tag	tag特征
tag_appr_flag	tag是否在query中
tag_convert	tag转化率

备注：转化率为加贝叶斯平滑十折有序计算的点击率

提升效果　初赛时线上A榜提升约2个千

- word share特征

统计词频，计算词的 tfidf 权值。针对 query 和 titile 的共有词，计算

tfidf_share: $\frac{share\_wights}{total\_weights}$

count_share_rate: $\frac{len(share\_words)}{len(q)+len(a)-len(share\_words)}$

num of share words: $len(share\_words)$

num of 2gram share words: $len(2garm\_share\_words)$

cosine_dis: $\frac{share\_w \bullet share\_w}{\sqrt{q\_w \bullet q\_w}+\sqrt{a\_w \bullet a\_w}}$

share_rate: $\frac{len(share\_words)}{\max(len(q),len(a))}$

9、 排序特征

每个特征内部也可以争对同一个query定义一个排名，尤其是相似度特征

9、 nn stacking特征

直接拿nn的某一嵌入曾输出作为lgb特征，拿esim来说，可以把maxpooling和averagepooling那层之后的作为lgb特征，这会是相似度特征的一个很好的刻画。

特征选择

每个特征计算耗时不同，重要性也不同，需要对他们进行一个统一的筛选，选择方案有大概下面三种（当然仅仅是TOP5方案中出现或者我自己使用的）：

- 直接根据lgb的feature_importance，这样做的特征，筛选，一般会使得性能有所降低
- 过滤法：相关系数、方差选择
- 包裹法：递归特征消除法

算法模型

机器学习

机器学习模型自然是以lgb为主的，lgb对数据不太敏感，它的性能高低完全取决于前面所做的特征，TOP5方案有人把lgb做到了0.626，最终也取得了较好的排名。

深度学习

深度学习主要就是esim模型，几乎所有队伍都得出一致的结论：在这题上面esim是最优nn，大家主要就是争对这一nn进行讨论。

对于esim，首先要讨论的就是要不要加入第三个输入，大部分组是直接选择加入手工特征来提高nn单模成绩，但这样做会降低nn和lgb的差异性，使得融合的收益减少。值得注意的是，第五名的队伍他们在nn中没有加入第三个输入，最终两个lgb（0.57， 0.59）和一个esim（0.58）融合出了0.627的成绩，也就是不加入第三个输入的nn融合收益很大。

接下来就说加入手工特征的nn，大部分队伍都是把lgb全部特征输入nn，对于我个人而言我是选取了lgb的部分特征输入了nn，我觉得加入太多特征会扰乱nn反向传播，会使得第三个输入反向传播的梯度比较大，第一个输入和第二个输入反向传播的梯度比较小，当然这仅仅是实验结论（打印nn的weights），但是这选取方法是纯手动的。esim融入手工特征有两种方案：

- concat方案
- gate方案（要略微优于concat）

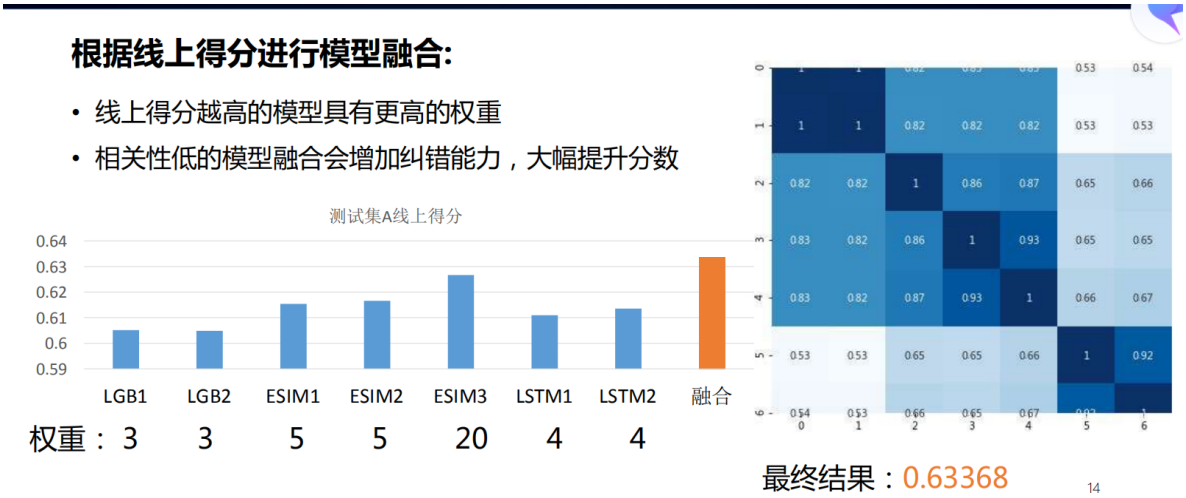
  eisim模型上的微调还有将两层BILSTM改成BIGRU和CNN进行融合，以此增加模型上的多样性，也就是即使对于同一组训练集，在不对模型做大的变动下仅仅改变那两层BILSTM就可以得到三个esim变体。对于我个人来说的话，我尝试过将esim的Embedding层进行替换，也就是我训练了两个词向量fasttext和word2vec，仅仅替换它们就可以得到两个esim变体。

  在nn训练时，有的队伍采用了CLR修改学习率，以及SWA得到全局化的通用最优解。

  最后说说自己队伍的一些在nn上做的主要变动吧，我们仅仅在手工特征的基础上加入lgb stacking特征（即五折lgb的预测值），就使得nn从0.602上升到了0.618，进入了前十

模型选择

下面是第三名的方案，用相关系数进行模型选择



模型融合

模型融合最主要的就是增加模型的多样性，在增加模型多样性方面有下列方案

- 在nn中不加入额外特征与lgb融合，这个收益是相当大的
- 对nn进行结构上的调整
- 选择不同的数据集，比如拿第一亿做一个模型，第二亿再做一个模型，第一名拿了6亿数据，我觉得这是他们组取胜的刚关键，他们是所有方案中利用数据集最多的组，对于nn，越多的数据性能越好。
- 如果你争对同一个数据集有多个nn，比如esim、孪生lstm，孪生cnn等等，可以尝试在多个nn的基础用了multi model合成一个nn 然后使用snapshot集成寻找单个multi model多个局部最优值来进行融合,这个操作确实很惊艳

其他思考

二分类（bce loss）问题和排序问题（pairwise loss）的思考，大部分队伍都是将这一题视为前一类问题，对于后面一类问题我自己不太懂，但是第二名方案的实验结果表明lgb排序loss的性能比lgb二分类loss的性能要出色。

最后回过头来看前三名的方案，他们都使用了3亿以上的数据量，而我们自己队伍仅仅使用了一亿的数据量，我感觉对于这一题的关键点就是数据量的使用吧。

参考

- TOP5方案的PPT
- 第三名完整解决方案
- 第五名完整解决方案
- 第九名完整解决方案，求star