**https://dc.cloud.alipay.com/index#/home**

**ATEC2018 NLP赛题总结**

今年5月份报名了蚂蚁金服的比赛，有金融大脑和风险大脑两个赛题，金融大脑主要解决智能客服遇到的自然语言处理问题，对于两个语句，判断是否是同一个意思，帮助构建客服的专用问答库，比赛的评判标准是f1分数，这对于正负样本不平衡问题比准确率更好，风险大脑则是通过用户登录和交易信息判断此次交易是否存在风险，在网络安全形势严峻的今天，其重要意义不言而喻。

选了金融大脑赛题，意外进入了复赛，2个月时间的投入，最终拿到18名，稍微总结一下，先把想到的写出来，后面想到什么再完善。

**题目简介**

语句对主要来自蚂蚁花呗的客户提问，语句是否是同种意思的标注来自外包团队，据官方介绍，根据抽查结果，95%的标记是正确的。   
举个例子，数据格式是句子id，两个句子，1个标签，初赛提供了10万个语句对，复赛50万。测评都是1万条。

id sent1 sent2 label

17 我开通不了借呗 我要申请借呗 0

18 借呗还款了，额度未恢复 借呗还款后额度没有恢复还显示借款 1

* 1
* 2
* 3
* 4

**我的方案**

训练多个不同匹配的模型，使用blending融合，简单说来还是模型堆砌，根据前排分享，对模型进行一定的改造还是能够增强效果的。

**效果简介**

主要从网上搜罗了四个不同模型，分别是自定义Siamese网络、ESIM网络、Decomposable Attention和DSSM网络，又按字符级和词级分别训练，最后一次的单模型的训练后在验证集上效果如下表

| **model name** | **模型输出与标签相关性r** | **最优f1评分** | **取得最优f1评分的阈值** |
| --- | --- | --- | --- |
| siamese char | 0.553536380131115 | 0.6971525551574581 | 0.258 |
| siamese word | 0.5308273808879237 | 0.6873517065157875 | 0.242 |
| esim char | 0.5853469280801447 | 0.7116622491480499 | 0.233 |
| esim word | 0.5783574742744366 | 0.7100964753080524 | 0.263 |
| decom char | 0.5288425401105513 | 0.6825720620842572 | 0.249 |
| decom word | 0.4943718720970039 | 0.6677430929314676 | 0.212 |
| dssm both | 0.5638034287814917 | 0.6980098067493511 | 0.263 |

前排最好的模型ESIM单模f1分数到0.72了，模型改造方面的能力还是存在差距的。

**其他收获，粗略记录如下：**

* 对keras的使用更加熟练，尤其是Callback的使用，
* 阅读了pytorch的文档，初学使用pytorch，其特点为：
  1. 强化版的numpy，前向运算和操作与numpy非常相似，而且可以直接利用GPU的运算能力。
  2. 与TensorFlow不同的是，pytorch无需编译图，每次backward都会根据当前运算过程构造新的图，然后销毁，在程序中甚至可以通过条件语句直接改变图的运行流程，启用或停止相关节点。
  3. pytorch对于最新研究成果的跟踪实现比keras快得多，拥有更丰富的神经网络层，更多优化器等。
* 学习了基于pytorch的fastai框架，框架的作者Jeremy Howard是Kaggle高手，fastai框架吸收了一些Keras中便于使用的特性，整个框架源码约4000余行，短小精悍，使用方便
* 跟着fastai的源码实践了统一语言模型精调(ULMFiT)方法，在文本相似度任务上并未取得好结果，ULMFiT方法特点如下：
  1. 训练一个语言模型，模型架构为Embedding + 三层双向LSTM(+dropout)，数据集一般为wiki，受限于数据加载和预处理方式，目前的源码仅能处理不超过500M的语料。
  2. 在当前任务语料上finetune语言模型。
  3. 根据当前任务设计分类器模块，其出入为语言模型最后一个LSTM层的输出，从最后一层开始，逐层unfreeze，进行分类器模型精调。
* 学习了batchsize参数对训练的影响，更大的batchsize意味着更准确的梯度方向，可以更快完成每个epoch，同时也意味着每个epoch的更新次数更少，需要更多的epoch才能使模型收敛，一味增大batchsize反而会延长训练时间。
* 学会使用循环学习率变化的训练技巧，Circular Learning Rate通过循环改变学习率，从小到大，从大到小，不断循环，使模型更容易跳出局部最优，做出更多的尝试，该方法确实调高了模型训练的结果。
* 学会使用SWA(stochastic weights averaging)模型融合方法，即将训练过程中的模型权重进行平均达到模型融合的目的，该方法的代价极小，仅需要保存另一份模型权重在内存或GPU显存中，在每个epoch后(或其他间隔)更新一次该权重，在训练结束时便可获得一个普通模型和一个SWA模型，该方法提高了5/7模型在初赛数据上的泛化能力，但并未提高任何模型对于5倍的复赛数据的泛化能力，这可能与模型训练不够充分有关。
* 学习了一些模型融合方法，包括求多模型平均、投票、Stacking和Blending模型融合方法，其中Quora比赛中的一个stacking方案值得借鉴，他们将训练数据分成5份，三份训练，1份验证，1份测试，轮番5次，直到每份数据都参与1次验证和1次测试，这比传统的stacking更好的利用了数据。
* 学习了语句对任务建模的两类基本模型，分别是向量表征模型和表征交互模型，向量表征模型利用孪生网络(Siamese Network)将两个语句编码成两个独立的向量，然后计算向量间的相似度，比如Siamese Net，DSSM；表征交互模型通过构造一个相关性交互矩阵，将两个语句的信息进行糅合处理，比如Decomposable Attention，ESIM。
* 除了ESIM外，其他人用了两个新模型DRCN和DIIN (DRCN是SNLI排行榜最佳模型）
* 本次比赛未尝试的方法：
  1. 利用句子的拼音作为辅助输入，通过拼音embedding加强模型，
  2. 将字、词和拼音等混入一个模型中，增强单个模型的能力
  3. DRCN模型
  4. **10折验证训练模型（大伙都在用）**