心得

本次是对整个项目的回顾。

首先是从数据入手，对于大量数据，当我们运行项目时候会造成很慢的，我们需要有优化的，数据的优化在其中占据很重要一部分，当数据量比较大时候，我们会用hadoop集群或者云服务的S3。然后从数据的pipeline上面开始优化，tensorflow中提供了一个dataset的库供我们调用.

Pipeline的原因

数据不能全部读入内存

同时需要预处理工作(cpu预处理)

更好利用硬件资源

分布式训练的

具体操作

Extract、transform、Load

正常情况下，cpu与GPU交替操作。我们可以从buffer\_size、num\_parallel\_calls将数据从串行设置到并行处理、interleave(提高io的能力)。

也可以使用tensorflow profile工具

标注的工具label-studio

然后从基础模型入手

RNN（LSTM-GRU）,seq2seq，针对其中权重重要性引入了注意力机制Attention。Seq2seq+attention到生成文本，这其中包括了生成策略（greedy Decoding(每次选择最好的 Beam search（每次选择前几），top-k采样（每次采样前几的），top-p(前百分几的采样））。

评价指标（rouge-N（n-gram）,rouge-L（最长公共子序列）,Rouge-S(skip-gram)）

在训练之前的一些策略

1.loss:在⽤很⼩的随机数初始化神经⽹络后，第⼀遍计算loss可以做⼀次检查(当然要记得把正则化系数设为0)。

2.接着把正则化系数设为正常的⼩值，加回正则化项，这时候再算损失/loss，应该⽐刚才要⼤⼀些。

3.试着去拟合⼀个⼩的数据集。最后⼀步，也是很重要的⼀步，在对⼤数据集做训练之前，先训练⼀个⼩的数据集，然后看看你的神经⽹络能够做到0损失/loss (当然，是指的正则化系数为0的情况下)，因为如果神经⽹络实现是正确的，

在⽆正则化项的情况下，完全能够过拟合这⼀⼩部分的数据。

参数调优

神经⽹络的训练过程中，不可避免地要和很多超参数打交道，需要⼿动设定，⼤致包括：

1.初始学习率 2.学习率衰减程度 3.正则化系数/强度(包括l2正则化强度，dropout⽐例)

对于⼤的深层次神经⽹络⽽⾔，我们需要很多的时间去训练。因此在此之前我们花⼀些时间去做超参数搜索，以确定最佳设定是⾮常有必要的。最直接的⽅式就是在框架实现的过程中，设计⼀个会持续变换超参数实施优化，并记录每个超参数下每⼀轮完整训练迭代下的验证集状态和效果。实际⼯程中，神经⽹络⾥确定这些超参数，我们⼀般很少使⽤n折交叉验证，⼀般使⽤⼀份固定的交叉验证集就可以了。

⼀般对超参数的尝试和搜索都是在log域进⾏的。

学习率

在训练中不断寻找最合适当前状态的学习率。

越⼤的batch-size使⽤越⼤的学习率

原理很简单，越⼤的batch-size意味着我们学习的时候，收敛⽅向的confidence越⼤，我们前进的⽅向更加坚定，⽽⼩的

batch-size则显得⽐较杂乱，毫⽆规律性，因为相⽐批次⼤的时候，批次⼩的情况下⽆法照顾到更多的情况，所以需要⼩的

学习率来保证不⾄于出错。

训练阶段

我们需要动态设置学习率，前期大后期变小

随着时间推进，训练集和验证集上的准确度都会上升，如果训练集上的准确度到达⼀定程度后，两者之间的差值⽐较⼤，那就要注意⼀下，可能是过拟合现象，如果差值不⼤，那说明模型状况良好。

Loss不下降的原因

1）数据的输⼊是否正常，data和label是否⼀致。

2）⽹络架构的选择，⼀般是越深越好，但是也分数据集。 并且⽤不⽤在⼤数据集上pre-train的参数也很重要的。

3）loss function 对不对。

Loss Nan或者无穷大原因

梯度爆炸造成Loss爆炸

检查输⼊数据和输出数据

损失函数可能不正确

你的Shuffle设置有没有乱动（我们使⽤了batch\_norm层，并且数据的分布极不规律）

训练之后问题

hard-negative

我们需要将负样本错误分类的样本反复训练，也就是不停的将困难样本拿去训练，让分类器更好地学习到难以学习的特征，简单来说就是熟能⽣巧。

遇到的问题

OOV

解决方法

Spell check、UNK、扩大词库、subword(BPE、wordpiece、ULM)、PGN

Word-Repetition

解决方法

不要重复的n-grams

增加loss项

PGN coverage机制

非似然训练和F^2 softmax

最重要的模型

Transformer和PTMs

PTMs

Bert（双向的transormer模型（mask LM NSP））

GPT(单向的transormer模型）

ELMO(双向的LSTM语言模型)

XLnet(双流自注意力）

Roberta(扩大训练数据、不要NSP、动态的mask、文本编码）