中国人工智能学会
Chinese Association for Artificial Intelligence

全球人工智能AI 2021技术创新大赛

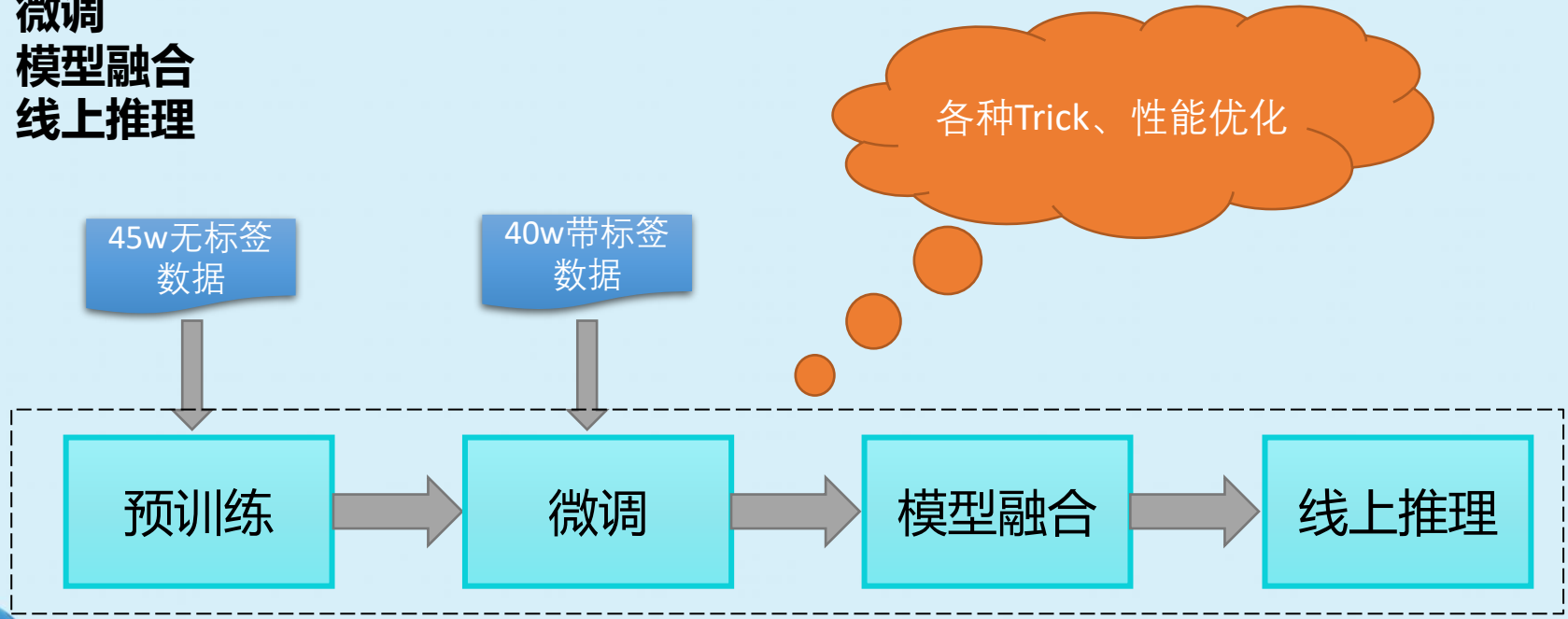
GLOBAL AI INNOVATION CONTEST

赛道三: 小布助手对话短文本语义匹配

ac milan

整体设计

1、预训练
2、微调
3、模型融合
4、线上推理



预训练-模型选取

- 赛题所给数据经过了脱敏，相当于一种新的语言，无法直接利用开源的预训练模型进行迁移学习
- 但是预训练依然很有必要，在有限的数据上，我们需要尽可能充分地利用其中的信息，Bert语言模型的MLM预训练任务可以利用无监督文本信息，学习文本表征、语言学知识和世界性知识
- 我们选用的是Bert和其变种Nezha，二者主要区别在于绝对位置编码与相对位置编码
- 考虑到后续的模型融合以及线上环境提供四卡，我们预训练了四个模型，参数量皆为1亿左右

模型类别	截断长度	词表大小与筛选词频	初始加载权重
Bert	32	18577 / 3	bert-base-chinese
Bert	100	13910 / 5	bert-base-chinese
Nezha	32	18577 / 3	nezha-cn-base
Nezha	100	13910 / 5	nezha-cn-base

预训练-MASK策略

我们对Bert的MLM预训练任务进行了改造，使用的是融入对偶的长度自适应动态N-gram Mask策略

- **模型输入为经典的拼接形式：[CLS] s1 [SEP] s2 [SEP]**
- **对偶：s1、s2以50%的概率交换位置，是对语义无损的数据增强方式**
- **长度自适应动态N-gram Mask策略**
  - 动态Mask：预训练达到400 epoch，上百万次iter，可以每次迭代都随机生成新的mask文本，增强模型泛化能力
  - N-gram Mask：以15%的概率选中token，为增加训练难度，选中部分以70%、20%、10%的概率进行1-gram、2-gram、3-gram片段的mask（选中token使用[MASK]、随机词、自身替换的概率和原版Bert一致）
  - 长度自适应：考虑到对短文本进行过较长gram的mask对语义有较大破坏，长度小于7的文本不进行3-gram mask，小于4的文本不进行2-gram mask
  - 防止小概率的连续Mask：已经mask了的文本片段，强制跳过下一个token的mask，防止一长串连续的mask

预训练-其他Trick与参数设置

- **学习率warmup与衰减**
  - 预训练400 epoch，前4.5个epoch，学习率从0线性增长到5e-5，之后线性衰减到1e-5
- **分块shuffle**
  - 预训练周期长，优化时间性能非常重要，分块shuffle将长度差不多的样本组成batch快，块间shuffle，减少padding部分运算量，耗时减少了约40%，实测不会降低模型效果
- **权重衰减**
  - 限制网络权值的大小，缓解过拟合现象
- **四个模型通用参数设置**

训练数据	初赛训练集10w；复赛训练集30w； 初赛A、B榜测试集合并5w；共45w
epoch	400
batch size	128
学习率	0 -> 5e-5 -> 1e-5
权重衰减率	0.01
随机种子	2021

微调-模型参数

➢预训练利用文本中的**无监督**信息，微调则需利用**有监督**的句子对**匹配信息**，将赛题任务建模为匹配与不匹配的**二分类问题**

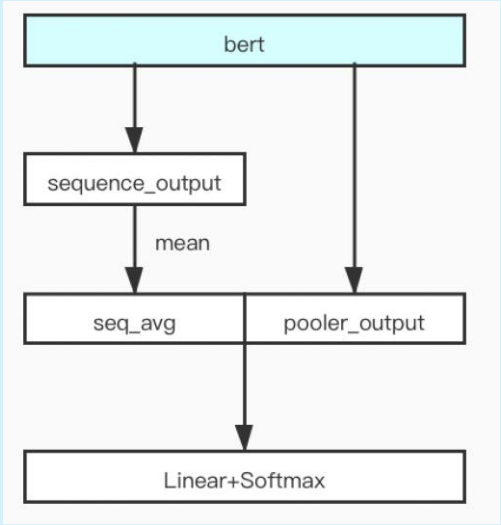
➢我们在**4个**预训练模型的基础上，训练了**6个**微调模型，从**词表**、**截断长度**和**模型结构**等维度保证模型之间的**差异性**，以便后序**模型融合**，参数设置对比如下：

预训练模型	截断长度	后接结构	学习率
nezha-base-vocab3-len32	32	最后一层向量取平均并与最后一层cls拼接	4e-5
nezha-base-vocab5-len100	100	最后一层向量取平均并与最后一层cls拼接	4e-5
bert-base-count3-len32	32	最后一层向量取平均并与最后一层cls拼接	4e-5
	100	最后一层向量取平均	4e-5
bert-base-count5-len100	100	最后一层向量取平均并与最后一层cls拼接	2e-5
	32	最后四层cls拼接	4e-5

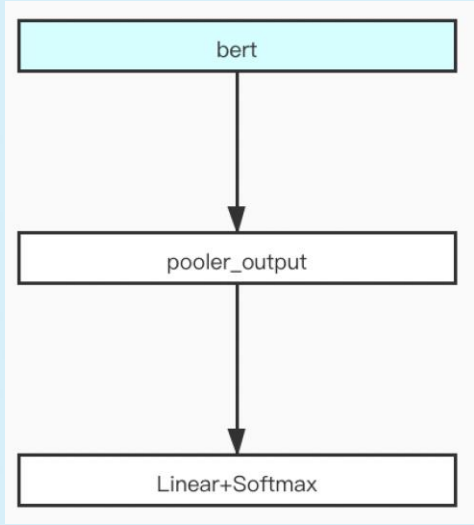
微调-后接结构

- **Bert/Nezha后接的三种结构**

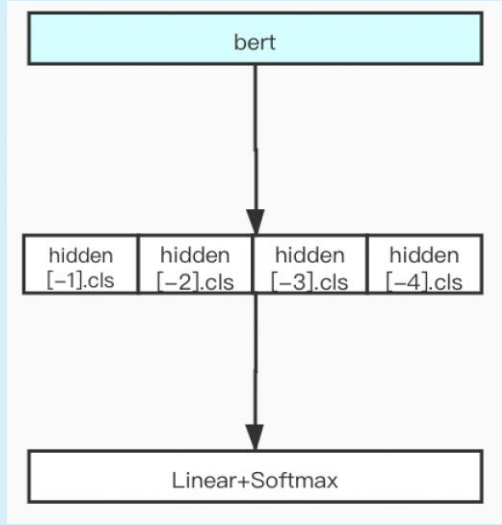
最后一层向量取平均并与最后一层cls拼接



最后一层向量取平均



最后四层cls拼接



考虑到Bert已经具备强大的特征提取能力，
以及运行和推理时限严格，所以其只后接了一些简单的结构。

微调-Trick

➢**学习率**

- warmup与衰减：可以使得训练初期学习率较小，模型可以慢慢趋于稳定，待相对稳定后再以预先设置的学习率进行训练，使得模型收敛速度变得更快。后采用学习率衰减的方式使模型收敛到更佳的极值点，提升最终效果
- 不同模型采用不同的学习率（2e-5或4e-5）

➢**模型融合时先对logits加权平均，后softmax**

使得softmax不再是每个模型独立进行，而是综合利用所有模型信息

➢**对抗训练**

对抗训练是一种引入噪声的训练方式，可以对参数进行正则化，提升模型鲁棒性和泛化能力

- Fast Gradient Method (FGM)：对embedding层在梯度方向添加扰动
- Projected Gradient Descent (PGD)：迭代扰动，每次扰动被投影到规定范围内

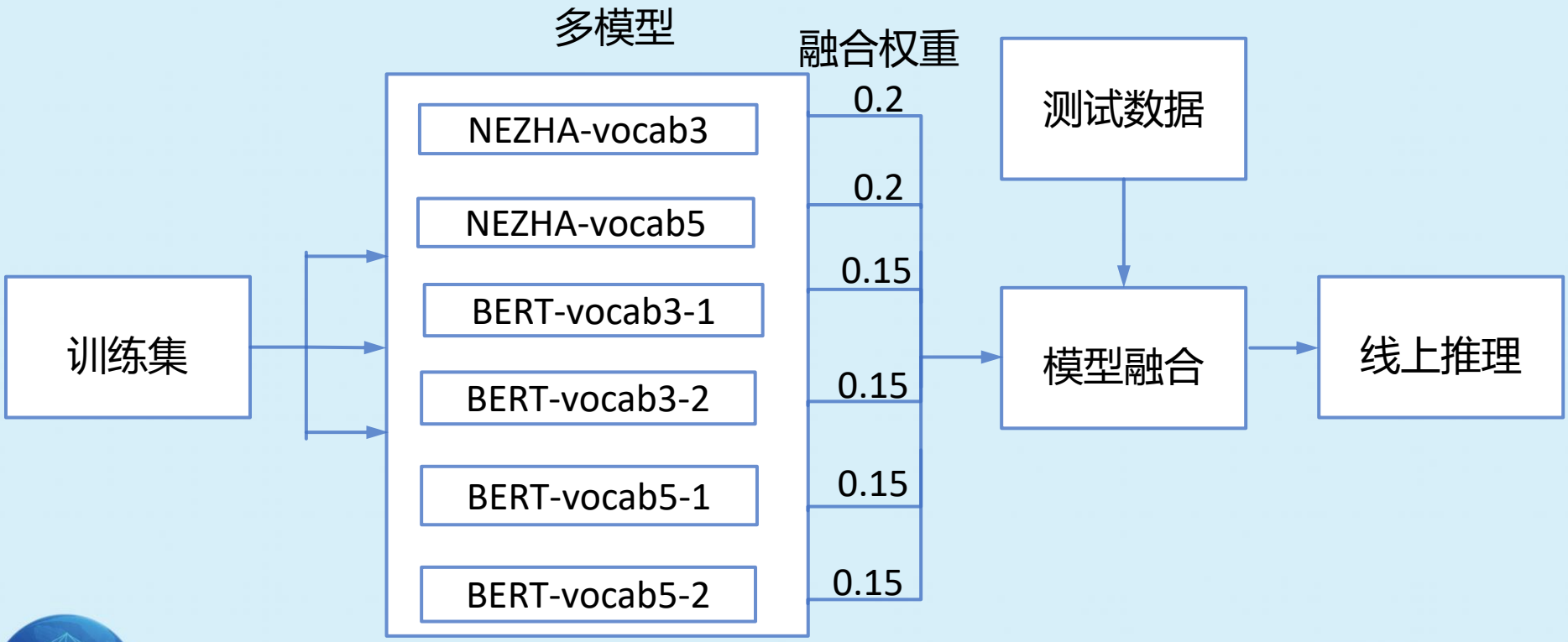
团队实验了FGM、PGD，前者速度快且效果更佳。

微调-通用参数

➢**最佳参数**

- batch_size=32，预训练充分的情况下，微调收敛非常快，小bs带来更大的随机性，更不容易过早陷入局部最优
- epoch=3
- dropout=0.2，训练时以一定概率丢弃某些神经元，缓解过拟合
- FGM，epsilon=0.25时效果最佳

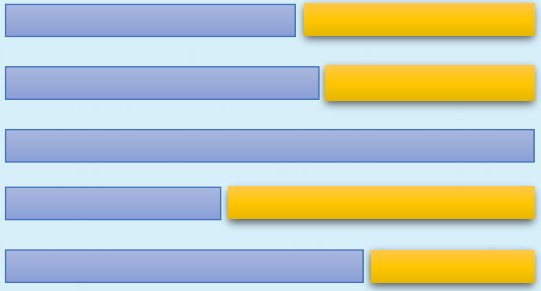
模型融合与推理


多模型
融合权重
NEZHA-vocab3
NEZHA-vocab5
BERT-vocab3-1
BERT-vocab3-2
BERT-vocab5-1
BERT-vocab5-2
0.2
0.2
0.15
0.15
0.15
0.15
训练集
测试数据
模型融合
线上推理

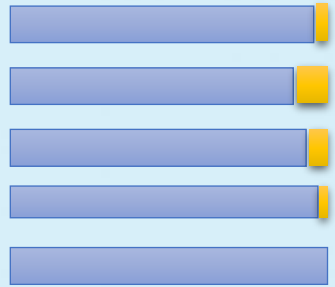
性能优化-分块shuffle

赛题限制线上总运行时间为80小时，限制推理5w测试集时间为15分钟（含网络开销），性能优化尤为关键

- **分块shuffle将长度差不多的样本组成batch快，块间shuffle，减少padding部分运算量，预训练耗时减少了约40%**
- **最终预训练线上能控制在9分多钟一个epoch，400个epoch能控制在65小时以内完成**



常用的整体shuffle，无法保证每个batch中数据长度差距不大，按最长数据进行padding（黄色标出），大量时间浪费在padding部分的计算上

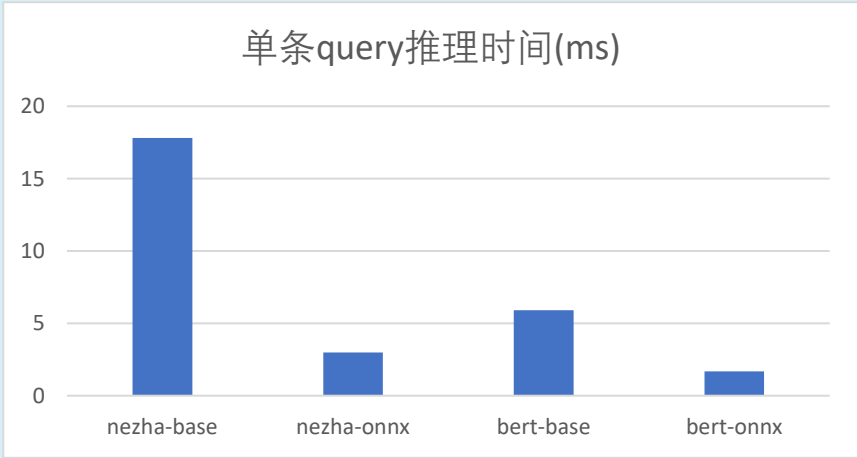


分块shuffle保证每个batch中数据长度差距不大，padding占比将大大减少，训练获得明显加速

性能优化-推理加速

➢**推理加速**

-**ONNX Runtime**：ONNX Runtime是机器学习模型的预测引擎，能使用内置的图优化（Graph Optimization）和各种硬件加速功能，来优化和加速推理。像BERT这样的Transformer模型，由许多运算符（Operator）的图构成，ONNX Runtime内置图优化功能，可以简化图并且减少节点，进而执行更复杂的节点融合和布局优化。通过使用ONNX Runtime，推理部分获得了非常可观的加速。



性能优化-对cuda版本的调优

➢在大家使用较多的cuda11镜像中，我们发现线上V100速度较慢，根据以往项目经验，老一些的卡用较新的cuda版本未必能发挥出最好的性能，我们尝试更换镜像版本为cuda10.2，cudnn版本配套改为7，onnxruntime-gpu版本配套改为1.5.1，推理速度有了较大提升，使得在15分钟内我们能跑6个模型（以往为4个）

模型类别	环境	推理5w条耗时
Bert	cuda11，onnxruntime-gpu 1.7.0	96s
Bert	cuda10.2，onnxruntime-gpu 1.5.1	84s
Nezha	cuda11，onnxruntime-gpu 1.7.0	220s
Nezha	cuda10.2，onnxruntime-gpu 1.5.1	150s

性能优化-其他细节

- **减少内存到显存的通信开销**：避免使用.to('cuda')的方式将tensor从内存移至显存，增加通信开销，而是一开始就用torch.tensor(xxx,device='cuda')的方式将tensor创建在显存

- **编写更快的分词函数**：所给数据已经用空格将token隔开，避免使用tokenize函数将数据整体当做字符串进行分词，而是按空格split后直接convert_tokens_to_ids

- ……

团队合作-Git的使用

- 赛题要求提交端到端可复现镜像，加之运行推理有严格时间限制，极大地考验了选手们的**工程能力**和**团队协作能力**

- 我们团队在码云建立了一个私有仓库，每个成员的每一次commit，都会由队友进行**细致的代码审核**，截至复赛结束，我们的仓库已经有了**4**个分支，共**79**次commit

- 部分commit记录

CONTENTS

创新

- 融入对偶的长度自适应动态N-gram Mask策略
- 不同词表、不同截断长度、不同结构的模型融合，保证模型差异性
- 学习率warmup与衰减、模型权重衰减、对抗训练等Trick
- 性能优化，包括分块shuffle、ONNX Runtime的使用、对cuda版本的调优和其他细节优化

落地

- 我们的模型将语义匹配转换为**分类问题**，这是一种**通用性非常强**的解决方案，可以广泛落地于自然语言处理领域中涉及到**句子关系**的各项任务中，如开放域意图识别（本赛题）、QQ匹配、QA匹配、文本蕴含等

- 推理速度较快，不计网络通信消耗，比赛使用的**6模**（4 Bert，2 Nezha）融合后可达**77**的QPS（AUC **0.9579**），在牺牲**不到一个百分点**的AUC下，**单模Bert**可达**595**的QPS（AUC **0.948**）

- 实际生产环境复杂，短文本相对容易出现**语义缺失**，且受噪声影响相对更大（用户输错或语音识别错误几个字，占短文本整体的比例可能就较大），可能需考虑辅以**指代消解**、**文本补全**、**文本纠错**等技术

- 深度学习并非万能，实际落地时，需要不断进行**badcase分析**，适当辅以**规则**的方法提升系统鲁棒性

CONTENTS

方案总结

总结性回答

我们从预训练、微调、模型融合和推理四个方面入手，每个阶段进行针对性的策略改进及创新，辅以性能优化，最终形成了一个较好的端到端解决方案，可以广泛落地于自然语言处理领域中涉及到句子关系的各项任务中，具有较好的实用性和创新性。

方法优劣势分析、展望

- 优点：效果好，速度快，模型通用性强
- 缺点：交互型模型因为每次计算都需要输入完整句子对，不适合于从海量文本中召回结果，而是适合在召回小部分候选集后，进行精细的排序
- 展望：从科学研究角度，我们要利用好预训练模型这个核武器，设计更有针对性，更加合理的预训练任务，此外也可探索结合上下文、引入知识的多轮匹配任务。从应用角度，可以从badcase出发，不断优化算法，挖掘用户需求，让小布成为一个知识更加渊博，对话更加流畅，更加人性化的智能助理


中国人工智能学会
Chinese Association for Artificial Intelligence
THANKS!