中国科学技术大学

CCKS2025-大模型生成文本检测评测论文

中国科学技术大学

CCKS2025 大模型生成文本检测报告

作者姓名：　莫环欣　袁则　崔玉皓

学　　号：PB22151796, PB23000298，PB22111663

学　　院：　CS、SGY、CS

第一章　模型修改历程

崔在本次比赛中主要负责调研工作，包括 QLoRA、数据增强、置信度方法等，最终在基准脚本的基础上同时应用伪标签、数据增强与置信度方法，获得了 0.8022 分，为本队本次任务的最高得分。同时还对 qwen3-1.7b 模型进行了尝试。但是限于硬件配置等问题，最终没能用上这个模型。

莫编写了 bert-base-uncased 基准脚本，并在此基础上尝试了 deberta-v3-large、deberta-v3-base 等预训练模型（即使有混合精度，但由于资源限制，表现不如 bert），调研 kaggle 相关竞赛发现了伪标签做法，应用后得分有个位数百分点的提高；并尝试在基准脚本上验证了由崔指出的置信度方法，在相同参数情况下得分由 0.6105 提高至 0.7078；另外尝试使用 BGE 向量化训练集并进行聚类分析，尝试通过多分类训练提升任务效果，但结果不佳。此外还通过调用百炼 API 测试了未经微调的 QWen-max 模型，但效果很差（据悉有参赛者仅在 QWen 上仅一轮简单微调即得到了 0.9 以上的评分）。

袁在本次比赛则中主要负责代码和实验，由于计算资源相对充足，完成了大部分对计算资源要求较高的实验（包括一部分 BERT 实验和全部 LLM 实验），并使用 LanguageTool 对原数据进行了处理。实验结果一度获得 0.7598、0.7698 分等较高的结果。并在本次比赛中贡献了最多的提交。

第一节　基本路线介绍

一、Bert

本次比赛中，我们首先使用预训练的 bert-base-uncased 模型编写了一个基准脚本，将目标设定为二分类任务，并结合模型特点对所有数据都进行了转小写处理，同时为了加快训练速度与应用更大的截断长度和批样本数，在训练器中加入了 torch 内置的混合精度逻辑。为了使得脚本更灵活（例如后续分层学习率等），这里自定义了模型的分类头，并在截断长度与批样本数都为 128 的情况下进行端到端训练。其在训练集与验证集上都很快收敛到较大值，但是在测试集上的得分仅为 0.6105，这一方面可能是截断长度的问题，另一方面（主要）也和比赛的举办方提到的数据分布的漂移有关，即训练集和测试集的数据分布存在较大差异。

二、LLM

起初我们采用了与 BERT 类似的方式，在 DeepSeek-R1 的 QWen-1.5B 蒸馏版前接上分类头进行训练，但是效果并不好。在官方群进行调研了之后我们尝试了使用 QWen0.5B、1.5B、1.7B、3B、7B 在内的一系列大模型，在这其中我们尝试了包括语法校正，QLoRA 微调，指令微调，置信度。等一系列的调参方法，可惜的是并没有取得什么明显的突破。最后我们队还尝试了 LLaMA-3.2 的 1B 版，但是同样效果低于预期。

第二节　对 BERT 的改进

在微调过程中，我们发现随着训练轮数的增加，测试集成绩出现明显下滑。这可能是以下两方面原因导致的：

- 训练集数据量与特征量太少，模型过拟合，泛化能力被削弱。
- 训练集本身质量不佳，导致模型难以学习到有效的特征。

针对这些问题，我们在后续实验中尝试了多种改进策略，包括伪标签的引入和数据增强的尝试，取得了一定的效果。

一、伪标签的引入

在 Kaggle 网站上的相关比赛中，我们调研了排名靠前的队伍所使用的方法，发现“伪标签”是许多队伍的常用策略。受此启发，我们将伪标签方法引入到本任务中。

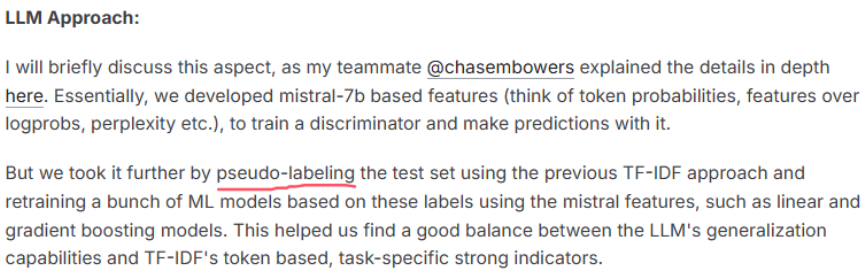


图 1.1　Kaggle 相关任务讨论区

具体而言，在对模型进行端到端训练后，我们首先对测试集进行预测并统计置信度分布情况，发现置信度大于 99% 的样本数占总样本数的一半。我们将这些高置信度样本加上模型预测的标签后，合并到训练集中。在其余条件相同的情况下，伪标签的引入使得模型性能较 BERT 的基准模型提升了 5%。

此外，为避免模型在原有训练集上过拟合而弱化新数据的作用，我们进一步

改进了训练策略：即将新数据和原有数据混合后，对重新初始化的预训练模型进行训练（而非在经过端到端训练后的权重上继续训练）。这一策略进一步将分数提高到了 0.74，为本队在前期取得了较好的成绩。

二、置信度的尝试

在探索过程中我们发现有些时候模型在训练过若干轮后会出现几乎全是 0 或几乎全是 1 的情况，这可能是模型在训练集上过拟合或学到了一些过于浅显的信息导致的。但是这不代表模型没有学到有用的信息，所以我们尝试了置信度策略。该策略通过提取答案所在 token 的 logits，计算 “yes” 和 “no” 两个 token 的 softmax 值，并将对人类文本预测置信度高于 0.1 的结果全部标注为人类文本。这种方法纠正了模型在有限数据集上学习到的过于浅显的知识，调整了模型的输出分布，使得模型的能力充分发挥，对于当前模型与当前训练参数而言起到了一定的正向作用。

(a) baseline　　(b) 引入置信度策略

图 1.2　引入置信度策略前后对比

三、数据增强的尝试

在实验早期，我们对数据增强策略在本任务中的作用持保守态度。我们认为，普通的增删改等微小扰动可能会严重破坏数据的原有特征；而同/近义词替换可能涉及 “外部知识引入”，不确定是否被允许。

然而，在实验后期，由于 LLM 路线上引入的语法校正模块表现较好，我们决定在 BERT 路线中尝试一些简单的数据增强策略。实验结果表明，这些数据增强方法在一定程度上提升了模型性能。

(a) 直接提交结果　　(b) 反转目标结果

图 1.3　数据增强 + 置信度

四、新模型的尝试：DeBERTa-v3

在成功引入伪标签后，我们进一步尝试了 Kaggle 上常用的 DeBERTa 系列模型（主要为 v3 系列的 base 和 large）。DeBERTa 模型在 Kaggle 的相关任务中表现优异，因此我们起初对其抱有较高的期望；不过由于资源限制，没能引入 pipeline。

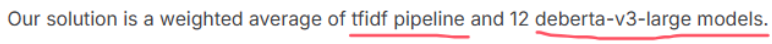


图 1.4　Kaggle 相关任务讨论区

然而，本次比赛实际应用中，DeBERTa 的效果非常差（大部分弱于 baseline）。由于 BERT 路线组员的资源限制，我们在 DeBERTa 模型的训练中只能进行分类头的微调任务，冻结远离分类头的权重，或者使用分层学习率配合极小的截断长度和批样本数（相比之下，bert-base-uncased 是端到端训练，这些限制可能是性能差异的主要原因）。最终，DeBERTa 路线的最优得分仅为 0.6 左右，该路线因此被废弃。

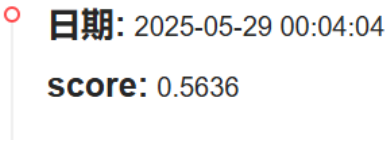


图 1.5　DeBERTa-v3-base+ 伪标签

五、更大胆的尝试：聚类 + 多分类

在过程中还尝试了使用 BGE 模型将训练集中的大模型生成文本向量化，并使用 sklearn 提供的接口对数据进行聚类，按照实际的模型数量 n 将整个训练集重新分为 n+1 类，并在这个 n+1 类的训练集上重新训练，将最终结果中的非 0 预测值全部改为 1 后提交。理论上来说如果各模型的生成文本具有不同的特征，这种方法应该可以将其区分出来，但是可能由于没有特别区分人类文本，导致这些特征中忽略了与人类文本最相关的部分；并且由于大模型的标签数众多，模型在随机分类时预测为大模型文本的概率也更大，种种因素叠加之下，最终表现不佳。

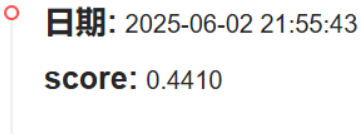


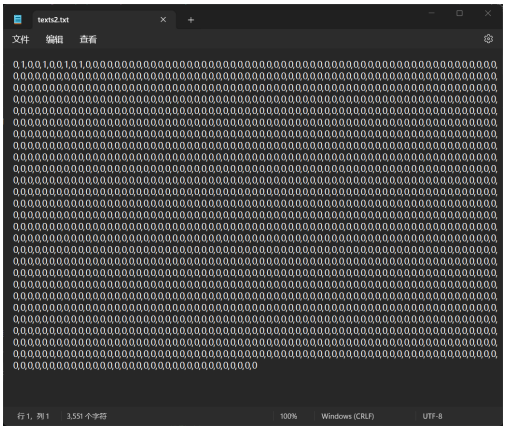
图 1.6　聚类后多分类

第三节　对 LLM 的改进

在初始几轮实验通过在 BERT 上接分类头训练的方法取得了不错效果之后，我们想将该方法运用在更大的模型上希望得到更好的效果。我们使用了 DeepSeek-R1 的 QWen-1.5B 蒸馏版，在最后一层的 CLS token 上接上一个分类头进行训练，并分为直接传递文本和添加一个 prompt 提示模型任务类别两种方式，可惜效果并不理想（如图1.8所示）。这可能是因为该聊天模型的预训练任务为文本续写，而非 BERT 的掩码语言模型任务，导致 BERT 的理解能力更强，更适合这样的分类任务，而 DeepSeek-R1 蒸馏版的理解能力较弱，无法很好地处理这种任务。

我们还使用了对没有微调过的大模型直接测试的方式，通过调用阿里百炼的 API 对 QWen 系列模型进行了直接评估，结果显示 QWen-max 和 QWen-plus 的效果都很差（如图1.7所示）。这可能是因为这些模型并没有经过针对性的微调，预训练时也没有掌握类似能力导致的。

(a) QWen-max



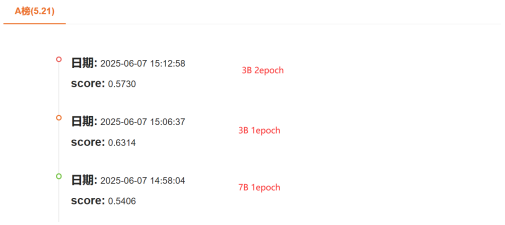
(b) QWen-plus

图 1.7　阿里百炼, 调用 API 直接评估测试集

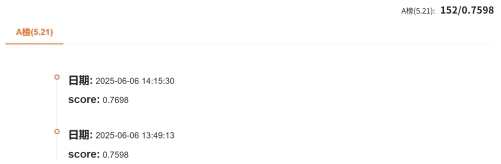
接下来我们尝试采用指令微调的方式，将待预测文本和 prompt 一起传入大模型，规定其作答格式，并通过关键词匹配来捕捉大模型答案。最初我们采用了较为简单的全量微调形式。对 Qwen-2.5-1.5B 模型进行了测试，发现模型极容易过拟合，且在测试集上表现不佳。我们推测可能是因为数据集数据量过小且训练集和测试集做了一定的分布切割，导致模型分布较差。浏览社区时我们发下了检

图 1.8　DeepSeek

测“空格加英文句号”关键词的 Baseline 也取得了较好效果，我们推测训练集和测试集中这种语法错误的不同可能会影响模型学习，故我们使用了 LanguageTool 对训练集和测试集都进行了语法校正，避免模型学习到较为低级的规律影响效果。



(a) 无语法校正数据



(b) 语法校正结果-2/3epochs

图 1.9　QWen-3B 模型

除了我们数据集的语法错误问题外，全量微调可能会对参数改变过多，而训练集数量和蕴含的知识都有限，故我们尝试了 QLoRA 微调方法。QLoRA 是一种低秩适配的微调方法，能够在不改变模型参数的情况下，通过添加低秩矩阵来调整模型的输出。我们使用了 QWen-3B 模型进行 QLoRA 微调，并尝试了不同的 epoch 数。

我们首先选择训练 4 个 epoch。然而得到的结果也不好（如图1.10b所示）。我们猜测可能是因为对原训练集过拟合的原因，所以我们接下来尝试了 2，3epoch 的结果，结果如图1.9b所示。可以看到我们这时取得了一个较好的结果，socre 达到了 0.75+。但是这个结果相较于先前的 BERT 只能说是持平。所以说明参数量上升，但是得到的效果并没有显著的上升。所以还是没有达到我们的预期。

此外发现大体量模型的微调效果似乎还不如小体量模型，例如图1.10a所示的 QWen-7B 模型的两次结果，并且长时间训练极有可能过拟合。

所以我们队就陷入了一个困境。一方面训练的 epoch 太少就会出现训练次数不足，难以拟合实际的数据集。另一方面训练的 epoch 太多又会出现过拟合。我们所有的模型都指出了一个问题。训练集本身并不难拟合，相反，想要拟合这个

(a) QWen-7B-语法校正　　(b) QLoRA-4epochs

图 1.10　QWen 模型

训练集实际上非常简单。大多数模型都可以迅速地收敛，loss 都可以很快地降低到很低的数量级。但是由于训练集和测试集的分布不同，这就导致了训练集拟合得越好，实际上在测试集的效果就越差。

我们按照这个思路自然是想要对数据进行一下泛化。但是由于在比赛规则中不得引入任何外部知识或生成完全新创的内容。这就限制了我们只能使用最传统的类似替换，裁剪等方式来对数据进行增强。

最后我们在后续调研还查到了与之前我们在 BERT 中所采用的类似的置信度的方法。在这种训练集和数据集分布不相似的情况下使用置信度的方法未尝不是一个思路，但是实际的置信度的调整还是一个经验调整。所以需要仔细调整超参数数值，调整不当就会出现全 0 或全 1 的情况。由于测试机会有限，我们也没有做足够多次数的尝试以取得较好效果。

第二章　最终代码和结果

本次任务中我们共有 34 次提交，本章中将展示其中的最佳成绩及对应的代码简析。

一、最佳成绩

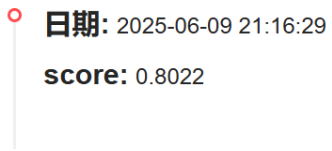


图 2.1　最佳成绩

二、最终代码

此处请参照我们实际提交的 Jupyter Notebook 文件中的代码块。需要注意的是最终的提交文件需要再将 0，1 反转一下。这样才能得到 0.8+ 的成绩。可以直接调用 reverse.cpp 来实现对结果的反转。最终交上去的 output.txt 就是已经反转好的，可以得到 0.8+ 的分数。

三、语法校正和数据增强

在这份代码里面我们使用了几个处理方法来提升我们的成绩。第 3，4 代码块中，我们使用了语法校正和数据增强。语法校正是让文本符合语法。数据增强则是通过同义词替换，随机插入，随机交换三种方法进行数据增强。虽然写了回译的处理方法。但是限于处理数据量较大和硬件限制，实际上并没有使用这个处理方法。

四、伪标签

伪标签是我们代码处理中的一个核心。在第 13-16 个代码块中我们，我们使用了为伪标签的策略。伪标签的策略核心是将我们非常确定的测试集的数据标注上我们的判断的标签并将其加入我们的训练集。也就是说我们新的训练集 = 原有训练集 + 伪标签数据。同时还有一个非常小的注意细节。我们在获得新的数据集之后并没有直接在现在以训练的模型上进行调整，而是将模型重新初始化之后在进行训练。

五、置信度

在最后我们使用了调整置信度的方法。在代码的最后我们的置信度选择是只要为 0 的概率大于 0.1 我们就选择 0. 但是我们实际上发现这样得到的分离非常低。于是我们将所有的结果进行反转。即 0 转 1，1 转 0。这样就能得到最终的结果。

第三章　总结和文件结构

一、总结

这一次实际上比赛还是难度颇大的，虽然我们组实现的最终成绩并不是很高，但是我们所有人都共同出力，可以称得上是问心无愧。我们都从这一次的比赛收获颇多。同时不得不吐槽一下，这一次我们查阅资料，后期加入比赛群才发现，群里的高绩普遍都在使用 ali 自己的 qwen 模型。但是我们队自身的硬件资源限制了我们对大模型的使用，希望以后的课程比赛可以选择限定算力的比赛。

二、文件结构

在提交的压缩包中我们包含了最终结果的截图，最优结果的提交 txt 文件”output.txt”，BERT 路线的代码 bert.ipynb,bert 路线对最终结果反转的 cpp 程序”reverse.cpp”，LLM 路线的代码 code 文件夹，实现语法纠错的 data_processing.py，最后还有论文本身。