上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

基于 ReLM 模型的中文拼写纠错

LogCreative

2024 年 5 月 25 日

摘　要

本文基于 ReLM 模型对 MAK 数据集进行中文拼写纠错，在测试集上达到 68.18% 的准确率。

目　录

1 数据集

MAK 中文拼写纠错数据集语料内容相近于小红书软件的说话风格，相较于普通的中文语料而言，其语料中包含有较多的自造词（“集美”、“一下子就看会惹”），这些自造词由于大概率不在词典中可能会被误判为错字，但是在该数据集环境下实际上是正确的中文表达，这是在该数据集上取得良好效果的一大挑战。

MAK 数据集包含两个部分：训练数据 train_data.tsv 有 5000 条输入输出对，测试文件 test_data.tsv 有 200 条输入数据。本文将训练数据按照 9:1 的比例顺序划分为训练集 train_data_2.tsv（4500 条）和验证集 dev_data_2.tsv（500 条）。

2 模型介绍

2.1 BERT

近年来，基于注意力机制的模型，特别是 Transformer[1]，由于可以训练出长距离的依赖信息、相较于普通的循环神经网络（Recurrent Neural Network, RNN）更加并行，而成为自然语言处理领域具有革命性的一派。其中，BERT (Bidirectional Encoder Representations from Transformers)[2] 基于 Transformer 通过上下文双向预测大幅改善了自然语言处理领域先在大数据集上预训练、再在特定数据集上微调方法（即 pretrain–finetune）于语言任务中的效果。

如图 1 所示，BERT 的输入是以分隔符分开的文本序列，输入序列可以是一句话，也可以是多个句子。每个输入序列开头使用一个特殊的 `[CLS]` 符号进行标记，每句的句尾用标识符 `[SEP]` 以标识和分隔不同句子。这里的 `[CLS]` 符号在经过 BERT 编码后能起到作为整个输入序列的上下文表示的效果[3]。

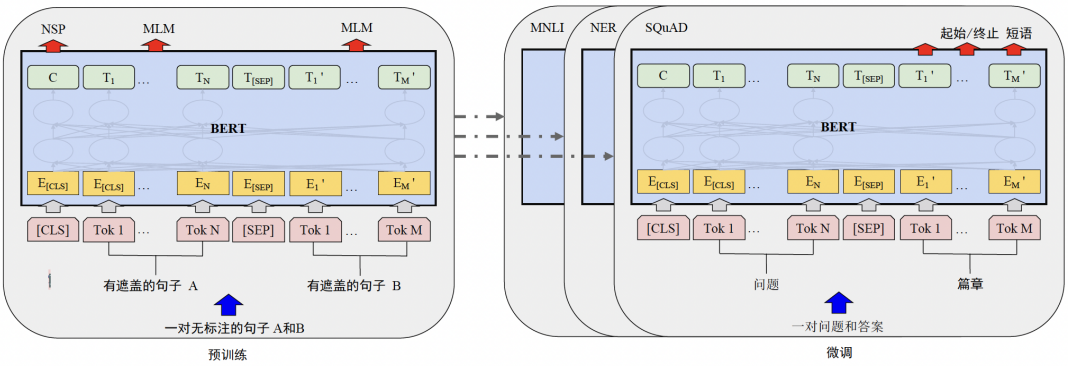

图 1 BERT 结构图[3]

bert-base-chinese[4]是谷歌官方基于中文语料库的 BERT 预训练模型。

2.2 ReLM

ReLM (Rephrasing Language Model)[5] 提出了新的纠错训练目标“重述语言模型”，相较于 BERT-Tagging，尽可能减少模型对于词对记忆的过拟合现象，并尽可能通过语义层面对错别字进行修正。

如图 2 所示，基于 BERT，ReLM 给出了一种非自回归（non-auto-regressive）的实现，相较于自回归中使用 [EOS] 来标识句子结束，ReLM 不使用该标识符号而是采用 $\{x_1, \cdots, x_n, \langle\text{sep}\rangle, m_1, \cdots, m_n, \langle\text{sep}\rangle\}$ 强制模型输出相同字符数目的文本。为了进一步减少学习到词对的可能性，并增加理解语义的可能，还会对输入随机遮掩，使得输入进一步变为 $\{x_1, \cdots, m'_i, \cdots, \langle\text{sep}\rangle, m_1, \cdots, m_n, \langle\text{sep}\rangle\}$。

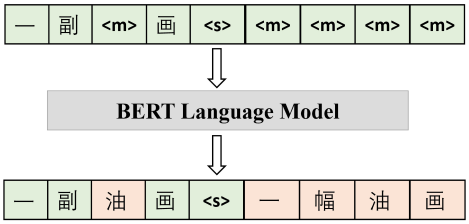

图 2 ReLM 的输入输出

3 模型选择与评估

正如第 1 节指出的，如果直接使用普通的 ReLM 预训练模型对 MAK 测试集进行推理，可能会对专有名词有很多的误判。因此，本文通过使用训练集 train_data_2.tsv 对 ReLM 预训练模型（基于 bert-base-chinese 并已在 3400 万中文语料上微调的预训练模型）进行微调，并在验证集 dev_data_2.tsv 上进行验证。

本文也尝试使用 Qwen1.5-7b[6] 作为基础模型使用 SFT 的方式进行基准实验。

验证集主要使用 F_1 分数作为判断标准，它是精确率 P（预测为正的样本中有多少是真正的正样本）和召回率 R（样本中的正例有多少被正确预测了）的调和平均数：

$$F_1 = \frac{2RP}{R+P} \tag{1}$$

测试集将使用 Kaggle 评测系统[7]得出准确率分数 ①。

4 性能分析

4.1 实验设置

实验操作系统为 Ubuntu 22.04，GPU 为 Nvidia A100 80G，CUDA 版本为 11.8。

对 ReLM 微调时，批量大小（batch size）为 128，AdamW 优化器初始学习率为 1×10^{-5}，预热比例（warmup proportion）为 0.06，权重衰减（weight decay）为 0，随机遮掩噪音概率（noise probability）

①由于写作时比赛尚未结束，所以该结果只基于 33% 的测试集。

为 0.2。在没有特殊说明的情况下使用 fp16 混合精度，每隔 100 个轮次（epoch）验证并保存模型，最大 1000 轮次。

对 Qwen1.5-7b 微调时，批量大小为 1，AdamW 优化器初始学习率为 3×10^{-5}，预热比例（warmup proportion）为 0.06，权重衰减（weight decay）为 0，最大 3 轮次。采用 fp16 混合精度。LoRA 配置[8]为

```
LoraConfig(
    r=8,
    lora_alpha=16,
    target_modules=["q_proj","k_proj","v_proj",
        "gate_proj", "down_proj", "up_proj"],
    lora_dropout=0.1,
    bias="none",
    task_type="CAUSAL_LM",
)
```

4.2 实验结果

如表 1 所示，ReLM 在随机种子为 1024 的情况下，学习率我为 1×10^{-5}，800 轮训练后在验证集上达到 92.20% 的 F_1 分数，在（33% 的）测试集上达到 68.18% 的准确率。

表 1 实验结果

序号	模型	训练集/验证集	随机种子	初始学习率	训练轮数	验证集 F_1 (%)	测试集准确率 (%)
0	ReLM+BERT	9:1	–	–	–	25.46	43.94
1	ReLM+BERT	9:1	42	5×10^{-5}	400	90.95	63.64
2	ReLM+BERT	9:1	2024	5×10^{-5}	400	90.72	65.15
3	ReLM+BERT	9:1	1024	5×10^{-5}	500	91.04	66.67
4	ReLM+BERT	9:1	1024	1×10^{-5}	800	92.20	**68.18**
5	ReLM+BERT	9:1	1024	5×10^{-6}	1700	**93.13**	63.64
6	ReLM+BERT	4:1	42	5×10^{-5}	600	89.43	62.12
7	ReLM+BERT	24:1	1024	5×10^{-5}	100	92.58	63.64
8	ReLM+BERT (fp32)	9:1	1024	5×10^{-5}	300	91.04	62.12
9	Qwen1.5-7b	9:1	1024	3×10^{-5}	3	$P = 60.00$①	28.79

4.3 参数选择

训练集/验证集划分比例 训练集越多，可用于训练的语料也就越多，但与此同时，验证集也就越少，模型验证时的准确率会有波动，影响对模型性能的判断。比较实验 1、6、7，可以看到 9:1 是一个比

①仅为句子准确率。

较适中的比例：4:1 没有足够多的训练数据；24:1 验证集过小，即使在验证集上拥有较好的性能，却没能在测试集上拥有较好的性能。

随机种子 由于“随机遮掩”用到了随机种子，所以随机种子也会对性能造成一定的影响。

初始学习率 初始学习率过大时，会导致跳过最优点的情况；初始学习率过小，会导致训练过程过慢，有时甚至到不了最优点。比较实验 3、4、5，可以看到 1×10^{-5} 是合适的学习率，更小时虽然在验证集上拥有更高的分数，但是在测试集上未必有更好的效果。

训练轮数 本文采用当前配置中 F_1 性能最高的对应轮数提交。

模型精度 大部分都使用了 fp16 混合精度，对比 3 和 8，可以看到使用原始精度 fp32 并不一定带来效果上的提升，而且训练时长和训练需要的显存大约增加了一倍。

5 结论

本文基于 ReLM 对 MAK 中文拼写纠错数据集进行实验，通过调整训练集/验证集划分比例、随机种子、初始学习率，最终得到在随机种子为 1024 的情况下，学习率我为 1×10^{-5}，800 轮训练后在测试集上达到 68.18% 的准确率。

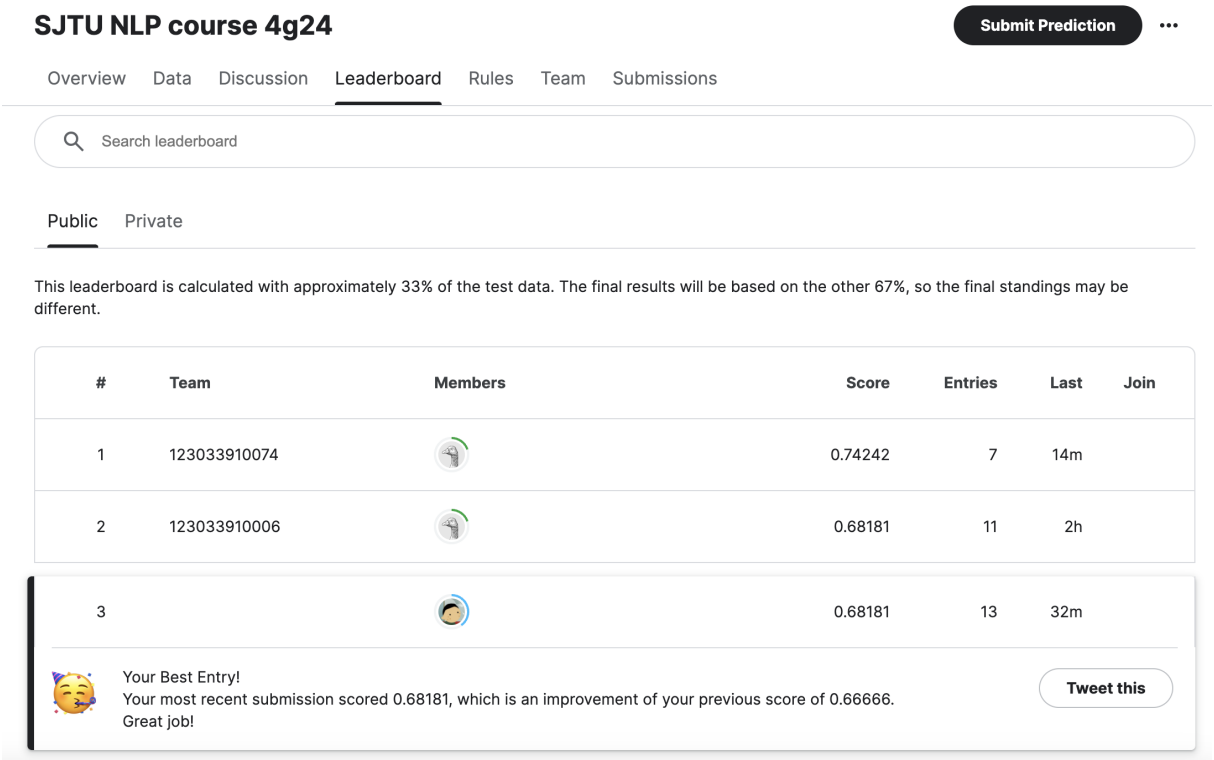

图 3 Kaggle 结果（2023-05-25）

参考文献

[1] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems (NIPS), 2017, 30.

[2] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]//Proceedings of the 2019 Conferenceof the North American Chapter of the Associationfor Computational Linguistics: Human Language Technologies: vol. 1. 2019.

[3] 赵海. 自然语言理解 [M]. 北京: 清华大学出版社, 2023.

[4] Google-bert. bert-base-chinese[EB/OL]. (2023-02-19) [2024-05-25]. https://huggingface.co/google-bert/bert-base-chinese.

[5] LIU L, WU H, ZHAO H. Chinese Spelling Correction as Rephrasing Language Model[C]//Proceedings of the AAAI Conference on Artificial Intelligence: vol. 38: 17. 2024: 18662-18670.

[6] BAI J, BAI S, CHU Y, et al. Qwen Technical Report[J]. arXiv preprint arXiv:2309.16609, 2023.

[7] XINBEI M. SJTU NLP Course 4G24[EB/OL]. (2024-04-24) [2024-05-25]. https://www.kaggle.com/competitions/sjtu-nlp-course-4g24.

[8] Macropodus. Qwen1.5-SFT[EB/OL]. (2024-05-17) [2024-05-25]. https://github.com/yongzhuo/qwen2-sft.

A ReLM for Chinese Spell Correction

The codebase is modified from lemon[②] to train, evaluate, and test on the MAK dataset.

MAK Dataset may not be publicly available at the time. If you have the dataset, please place it in `data/mak` folder, and split the dataset by `train_data.tsv`, `dev_data.tsv` and `test_data.tsv`. And then using `python sft_csc_gen.py --input [file] --output [file]` to generate data in `data/mak_sft` folder in `train_data.jsonl`, `dev_data.jsonl` and `test_data.jsonl`.

A.1 Usage

The repo is tested on Ubuntu 22.04, Nvidia A100 80G GPU.

1. Install Python and PyTorch with CUDA.
2. Install the dependencies.

```
pip install -r requirements.txt
```

②https://github.com/gingasan/lemon

3. Prepare pre-trained model[③] (from the original lemon[④] repo) in `output/relm-m0.3.bin`. Before the next step, you could also set the HuggingFace mirror to speed up the base model downloading by

```
export HF_ENDPOINT=https://hf-mirror.com
```

4. Finetune ReLM (training)

```
python run.py --model_type relm \
    --do_train \
    --do_eval \
    --load_state_dict output/relm-m0.3.bin \
    --fp16 \
    --output_dir output/relm-1e-5
```

This will use the default hyperparameter for training: `--train_batch_size 128 --eval_batch_size 128 --learning_rate 1e-5 --max_train_steps 1000 --seed 1024`. You could reduce the batch size to reduce the vRAM usage.

In the other terminal, open tensorboard

```
tensorboard --logdir output --port=12333
```

5. Test ReLM for submission

```
python run.py --model_type relm \
    --do_test \
    --load_state_dict output/relm-1e-5/step-800_f1-92.20.bin \
    --output_dir output/relm-1e-5
```

The output csv will be in `output/relm-1e-5/submission_relm_xxxxx.csv`. The load_state_dict may vary a little bit on the F1 score.

A.2 Report

The report (Simplified Chinese) is in file[⑤].

③https://drive.google.com/file/d/10vvkG_jzNK-CjIwlSvizhE1IOpnn9OqN/view?usp=share_link
④https://github.com/gingasan/lemon
⑤report/csc.pdf