S&A 队 CTC2021 比赛技术评测报告

章岳[1]，包祖贻[2]，章波[2]，李辰[2*]，李嘉诚[1]，李正华[1*]

1.苏州大学 计算机科学与技术学院，江苏 苏州

2.阿里巴巴 达摩院，浙江 杭州

(*通信作者电子邮箱 puji.lc@alibaba-inc.com，zhli13@suda.edu.cn)

摘 要: 本文描述了我们在第三届中国“AI+”创新创业大赛中文文本纠错比赛(CTC2021)中提交的参赛系统。CTC2021 是首个针对汉语母语者文本错误的纠错评测，因此具有数据稀缺、错误复杂多样等难点。在模型方面，我们提出了一个三阶段纠错系统，渐进式地解决文本中的拼写错误、语法错误、语病错误。此外，我们也使用了一系列有效的性能提升技术，例如基于混淆集的解码约束技术和模型集成技术等。而在数据方面，我们提出了一种基于规则的数据增强策略，生成了大量与汉语母语者错误文本尽可能相似的训练数据，缓解了数据不足问题。最终，我们提交的系统在测试集上的错误检测 F_1、错误纠正 F_1、总得分均位列第一。

1 任务介绍

伴随着互联网时代的到来，以人工方式进行文本校正面对海量数据的积累变得越来越难以实现。因此，文本纠错(Text Correction)任务逐渐成为了自然语言处理领域的重要下层任务。传统的中文文本纠错任务，如近年来 NLPCC2018 语法纠错评测[1]和 CGED2020 评测[2]，大都关注汉语学习者作文中存在的错误，而忽略了汉语母语者常犯的错误。CTC2021(Chinese Text Correction, CTC)中文文本纠错大赛是首届以汉语母语者撰写的网络文本为评测数据的评测任务，其纠错内容包含拼写错误、语法错误、语病错误等，从多个层面充分考察了现有中文纠错系统的能力，对政务公文、新闻出版、教育等领域纠错技术的发展具有很大的促进作用。

本次比赛共涉及 3 大类，共 7 小类错误，如表 1 所示。其中，语病层面的错误在目前的中文文本纠错任务中尚属首次涉及，给现有的中文纠错系统带来了不小的挑战。

表 1 CTC2021 错误体系

错误大类	错误小类
拼写错误	别字
	别词
语法错误	缺失
	冗余
	乱序
语病错误	语义重复
	句式杂糅

本次比赛的输入输出样例如表 2 所示，输入为可能含有错误的句子，输出为纠正的编辑操作，包含错误起始位置、错误类型、错误字词和修改答案。当系统预测的错误起始位置和错误字词与标注答案一致时，即可算作是错误检测成功；如果修改答案也一致，那么可以算

作是错误纠正成功。训练数据方面，官方提供了使用规则生成的约 30 万条伪数据。测试数据方面，资格赛阶段，比赛官方提供了 200 句的开发集和 972 句的资格集以供参赛队伍进行提交测试。决赛阶段，需要参赛队伍在 CodaLab[1]网站上提交可运行的系统进行评分。

表 2 CTC2021 输入输出样例

句子编号	输入句子	输出编辑
0011-1	第三局比赛俄罗女排的气势被完全压制，中国女排就此以一场 3 比 0 零封取得五连胜。	7, 缺失, , 斯,
0069-1	高速公路上交通事故的主要原因是司机违反交通规则或操作不当造成的。	29, 句式杂糅, 造成的, ,

综上，我们认为本次比赛的难点主要包括：(1)汉语母语者所写作的文本与汉语学习者的文本存在着较大的领域差异，无法照搬以往汉语学习者文本纠错评测中的优胜策略；(2)首次涉及了语义层面的错误，例如语义重复错误和句式杂糅错误，没有可供参考的前人解决方案；(3)缺少人工标注的领域内训练数据集。

2 数据分析

本次比赛，官方共开放了 2 批人工标注的评测数据，分别为开发集(200 句)、资格集(972 句)。其中，官方在决赛阶段提供了资格集的人工标注答案，因此，本节我们主要针对资格集(CTC-Qua)进行一些统计和分析。

首先是数据来源方面，本次评测的数据集全部来自从互联网上爬取的汉语母语者文本，与传统的汉语学习者数据集文本纠错相比，其行文风格、话语主题较为多样，话语较为流畅，很少出现大范围的不通顺。表 3 展示了本次评测资格集中的一些真实句子。

表 3 CTC-Qua 示例

编号	句子	类型
00204	人之所欲，生甚矣，人之所恶，死甚矣，然而人有从生成死者，非不欲生而欲死也，不可以生而可以死也。	文言文
00207	把这视为不祥征兆的阿加特闷闷不乐，快乐的安琴劝她勿要自寻烦恼。	小说
00225	2017 年 11 月 14 日，网络购物“双十一”过后，南京一高校校园内的快件派送高峰。	新闻
00539	你以后要独立生活，爸不能一直陪在你边。	对话

其次是错误类型分布方面。我们对官方提供的人工标注答案进行了统计，官方答案共标注了 6 种错误类型，共 520 个错误。其中，别字/别词类错误最多，共有 262 个，占比超过了 50%，此类错误大都是输入原因导致的拼写错误以及字词含义混淆导致的误用错误。其次较为频繁的错误是缺失错误，这类错误大都是因为用户输入时不小心遗漏了字或者词。值得注意的是，语义重复和句式杂糅两类语病错误分别有 73 和 27 个，占据了约 20%的错误，

[1] https://codalab.org/

给本次评测任务增加了不小的难度。语义重复指的是相同含义的词语在句子中重复出现；而句式杂糅则是一个句子混用了多种句式，导致了表达的混乱。此外，还有冗余、乱序等多种类型的错误，都需要参赛队伍解决。各错误类型的分布统计如图 1 所示。

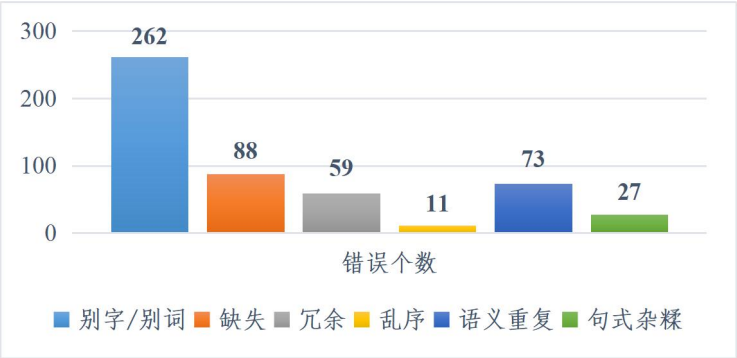

图 1 CTC-Qua 错误类型分布

此外，为了详细分析本届评测的汉语母语者文本错误数据集和以往汉语学习者文本错误数据集的差异，我们使用了 Charles 等人[3]开发的字级别编辑抽取工具 ERRANT-ZH 对 CTC-Qua 以及 NLPCC2018 测试集进行了分析。NLPCC2018 数据集来自于 PKU Chinese Learner Corpus 语料库，为北京大学搜集的外国留学生作文数据，并且由两位汉语母语者进行了错误标注。

表 4 CTC-Qua 和 NLPCC2018 测试集统计指标对比

语料名	句子数	错误率	字数/句子数	错误数/错误句子数	错误字数/错误数	纠正字数/错误数
CTC-Qua	972	49.59%	48.90	1.19	1.11	1.09
NLPCC2018	2000	99.15%	29.65	2.01	1.03	1.35

由上述表格可以看出，CTC-Qua 数据集中的文本错误远比 NLPCC2018 测试集稀疏，这一点体现在：(1)CTC-Qua 数据集中错误句子的比例远小于 NLPCC2018 测试集；(2)CTC-Qua 数据集中错误句子包含的错误数目也远少于 NLPCC2018 测试集。这主要是因为汉语母语者本身对语言较为熟悉，很少会在一个句子中出现多次错误，而汉语学习者对语言较为陌生，犯错更为频繁。而在句子长度方面，CTC-Qua 的平均句长约是 NLPCC2018 的 1.7 倍，因此对纠错系统处理长句子的能力要求更高。在每个错误涉及的错误字数以及纠正字数方面，CTC-Qua 和 NLPCC2018 两个数据集基本相近。

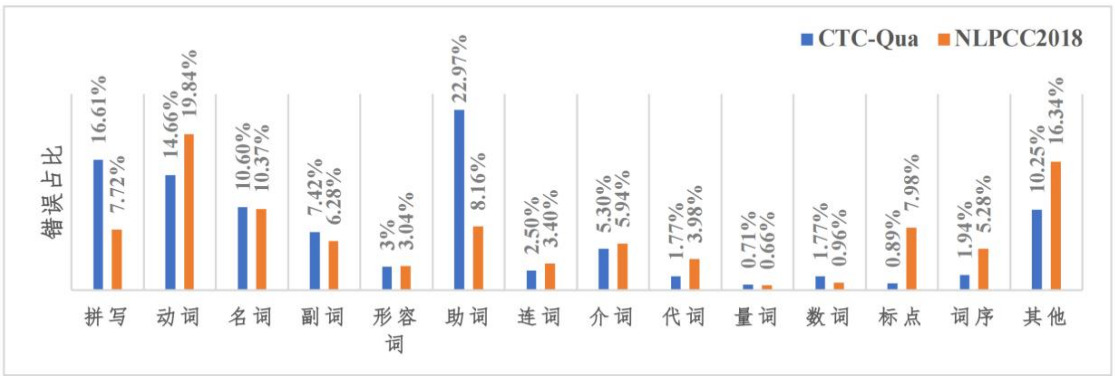

图 2 CTC-Qua 和 NLPCC2018 测试集错误词性类型对比

更进一步的，我们对比分析了 CTC-Qua 和 NLPCC2018 数据集错误类型分布的异同。

我们对 ERRANT-ZH 工具进行了扩展，使其可以支持词级别的编辑抽取和评价，将错误按照所对应的纠正答案词性(基于 LTP 工具[1])进行分类。对于替换类型的错误，我们使用爱奇艺 FASpell 工具中的音形相似度模块[2]判断其是否为拼写错误。最终结果如 2 所示，可以看出，汉语母语者(CTC-Qua)最常犯的两类错误为助词错误(如“了”、“的”等)和拼写错误，这主要是因为汉语母语者在互联网上写作时的疏忽导致的。而汉语学习者(NLPCC2018)犯动词错误、标点错误和词序错误的比例远远高于汉语母语者，这可能与他们对词义、语法、标点的使用掌握不牢有关。对于涉及多个词语的修改，我们统一归类为“其他”错误，而在这一类错误上，汉语学习者犯错的比例也明显高于汉语母语者，说明他们经常会出现多个词语的表述有误。

3 系统介绍

本次比赛，我们采用了一个三阶段纠错系统，针对性地解决不同类型和难易程度的错误：(1)我们首先使用基于序列标注的拼写纠错模型解决句子中的拼写错误；(2)然后将所得结果传入基于序列到编辑的语法纠错模型，解决剩余较难的语法错误；(3)对于涉及到语义的高级错误(如语义重复和句式杂糅)，我们使用基于语义模板的规则模型进行处理。

3.1 拼写纠错模型

针对拼写错误，我们使用了基于 BERT[4]的序列标注模型加以解决。当将 BERT 应用到拼写纠错任务时，我们利用 BERT 获取句子中每个字符的语义向量表示，将其传入一个全连接分类器，输出端的词表为常见字符。我们也额外尝试了其他最新中文拼写纠错模型，在综合衡量了速度和性能后，依旧选择了 BERT 加全连接层的方式进行本次评测的拼写纠错。

在模型训练方面，我们采用了基于混淆集生成的大规模人造语料预训练加真实数据微调的方法。此外，我们发现：Google 提供的 BERT 官方词表中缺失了一些常见的中文标点和汉字，因此我们又挑选了一部分常用标点、汉字对其进行了补全。

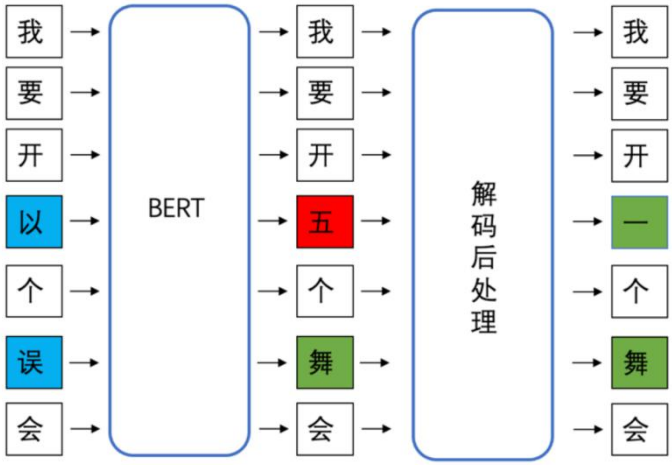

图 3 解码后处理示意图

在解码方面，我们发现拼写纠错模型可能会出现不恰当的修改，甚至会改变句子原义。针对上述问题，我们在模型解码时引入字音信息和混淆集信息作为约束，使得模型在每个位置不但要考虑候选字生成的概率，也要考虑是否和原字匹配。该算法具体流程为：针对原始

[1] https://github.com/HIT-SCIR/ltp
[2] https://github.com/iqiyi/FASPell/blob/master/char_sim.py

句子中的每个位置，挑选出所对应的 10 个概率最大的候选字符；按照概率大小依次遍历候选字符，如果其字音与原字符不相似且也不在原字符的混淆集中，则继续向后遍历，直到遇到符合条件的候选字符并选取其作为答案。如果遍历过程中遇到了[PAD]、[UNK]或者原字符本身，那么就保持原字符不变。如果遍历结束时依旧没有找到满足条件的字符，同样也保持原字符不变。图 3 展示了引入解码后处理策略带来的提升：当原字符为“以”时，相较于 BERT 模型直接输出的答案"五”，引入解码后处理策略可以选择读音上更为相近的正确答案“一”。

3.2 语法纠错模型

针对涉及到插入、删除等修改的语法错误，我们使用了目前基于序列到编辑的最优语法纠错模型 GECToR[5]进行解决。GECToR 模型本质上也是一个序列标注模型，它的解码空间是插入、删除、替换等编辑操作。通过并行预测编辑并将其应用于原句子，GECToR 模型能够完成长度可变的语法纠错。上述过程也可以多轮迭代进行，从而进一步提升修改效果。

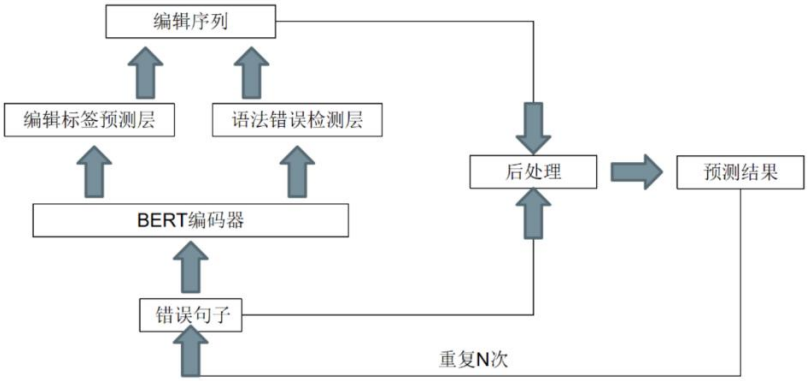

图 4 GECToR 模型推理流程

为了生成模型所需的训练数据，我们首先要从平行句对中抽取出编辑操作。我们参考英文纠错标签，设计了一套对应的标签体系。特别地，为了解决词序错误，我们设计了一个新的转移标签$MOVE。我们从人工标注的两个大规模语法纠错数据集 HSK 和 Lang8 中，抽取出现次数在 5 次以上的标签，并且过滤了其中涉及特殊符号的标签，共得到 8593 个标签。所有标签的具体统计信息如下表所示。

表 5 GECToR 模型编辑空间

标签	描述	个数	总个数
$KEEP	保持当前字不变	1	8593
$DELETE	删除当前字	1	
$APPEND(*t*)	在当前字后面新添一个字 *t*	4365	
$REPLACE(*t*)	将当前字替换为另一个字 *t*	4210	
$MOVE(*offset*)	将当前字移动到相对位置为 *offset* 的字之后	16	

接下来，我们需要将平行句对转换成标签序列。对于这一步骤，我们编写了基于最小编辑距离算法的标签抽取方法，并对其进行了中文适配。表 6 展示了编辑标签的抽取样例。

表 6 编辑标签抽取示例

原始句子	每天漫漫得进步。
纠错结果	每天都慢慢地进步。
编辑标签	每$KEEP 天$APPEND_都 漫$REPLACE_慢 漫$REPLACE_慢 得$REPLACE_地 进$KEEP 步$KEEPP 。 $KEEP

3.3 基于语义模板的语病纠错

针对涉及到较多语义知识的语病错误，例如语义重复和句式杂糅，现阶段由于训练数据的匮乏，所以此方向上的深度学习模型探索较少。对于此类错误，目前大都采用词语搭配模板[6]的方式进行纠错。因此，我们采用了基于语义模板(Semantic Pattern, SP)的规则匹配方式对上述错误进行纠正。

首先，我们利用网络爬虫，从大规模互联网语料中根据人工编写的特定模式自动抽取语义模板，并将其转换成为正则表达式。例如，根据模式“A。。。B 是病句吗”可以从“根据。。。显示是病句吗”这一句子中抽取出语义重复的两个词语“根据”和“显示”，从而可以转换成“根据.*显示”的正则表达式。利用上述正则表达式，我们就可以在句子中定位到含有语病的片段。

接下来，我们需要对找到的语病片段加以改正。针对语言习惯的特点，我们共定义了三种修改方式：(1)删除左侧词语；(2)删除右侧词语；(3)任选一侧删除。我们使用了基于语言模型困惑度的方式，来自动为每个语义模板选择修改方案。具体流程为：针对每个语义模板，我们用其匹配一批数据，在这批数据上计算删除左侧词语和删除右侧词语后的平均困惑度降低程度，选取降低程度较大的方案作为该模板最终的修改方式。如果删除左右两侧的对困惑度的影响很接近，那么就说明该模板可以任选一侧删除。最后，我们再对获得的模板进行简单的人工检查和修正，共积累了语义模板 875 个。表 7 展示了我们使用的一些典型语义模板。

表 7 语义模板

模板	修改方式
每天.*日理万机	删除左侧词语
原因是.*导致的	删除右侧词语
大约.*左右	删除两侧均可

3.4 模型集成及后处理

模型集成(Model Ensemble, Ens)：模型集成策略是一项重要的性能提升技术，将多个差异较大但性能略差的模型进行正确的组合，可以大大提升模型的准确性和健壮性。在本次评测中，我们使用了基于概率平均的模型集成策略，即将多个模型输出的概率取平均后视作最终的概率分布，以此来选取结果。我们分别对拼写纠错模型和语法纠错模型进行了集成，获得了稳定的性能提升。

后处理(Post-Processing, PP)：此外，我们观察模型输出的结果发现，有一些错误纠正可以归纳出典型的规律。因此，我们使用纠错系统对一份较为干净的语料进行了推理，对其中出现频率较高的误纠进行了规则干预，从而进一步提升了模型纠错的准确率。对于召回度较高但精确度较低的模型，我们还对保持原来字符不变的标签添加了额外的置信度，使模型倾向于做出保守的纠正。

4 训练数据构建

4.1 人造伪数据

本次评测任务的一大难点就是缺少领域内(即汉语母语者)的文本错误数据集可供使用。而基于深度神经网络的文本纠错系统，通常需要使用大量平行数据训练，才可达到较好的效果。因此，如何自动生成大量符合本次评测测试集分布的数据，将成为决定模型最终的性能

的一个重要因素。为了解决上述问题，我们尝试从语言风格和错误分布两方面去生成尽可能仿真的数据。

在语言风格方面，我们选用了一些尽可能和测试数据同源的母语者写作的中文单语语料作为加噪的种子数据，例如微信公众号语料库[1]和中文新闻语料库[2]。

表 8 加噪规则

粒度	操作	具体类型
词(50%)	替换(70%)	误用(10%)：随机从大词林[3]的近义词中采样替换
		拼写(90%)：随机从词级别同音混淆集中采样替换
	删除(15%)	缺失：随机从当前单词中删除 1 个字
	添加(10%)	啰嗦(50%)：将当前单词复制并插入到附近某个位置
		语义重复(50%)：随机从大词林的近义词中采样插入
	乱序(5%)	词序：随机将当前词语与附近某个词语交换位置
字(50%)	替换(70%)	形近(10%)：随机从字级别形近混淆集中采样替换
		音近(90%)：随机从字级别音近混淆集中采样替换
	删除(15%)	缺失：删除当前字
	添加(10%)	啰嗦：将当前字复制并插入到附近某个位置
	乱序(5%)	词序：随机将当前字与附近某个字交换位置

而在错误分布方面，我们主要尝试了基于规则的加噪方式。我们首先对原句子进行分词，然后对每 10 个词的片段随机挑选 1 个位置植入错误，这主要是为了保留错误位置的上下文语义，以及更符合汉语母语者文本错误稀疏的特点。加噪过程中，一共需要进行 3 次采样操作，分别确定加噪的粒度、操作和具体类型，采样的具体概率分布依据从 CTC-Qua 中统计得到的近似分布确定。表 8 展示了我们使用的具体加噪规则和采样概率。

其中，替换操作需要使用到 3 份混淆集，分别是：词级别同音混淆集、字级别音近混淆集和字级别形近混淆集。词级别同音混淆集的构建方法为：从微信语料中获得所有中文词语，并根据拼音是否相同对它们进行归类，与此同时筛除低频词。字级别音近混淆集的构建方式与词级别同音混淆集类似，但考虑了模糊音(如“zhang”和“zang”)，并且过滤了生僻字。字级别形近混淆集的构建方法，主要依据的是 FASpell 工具中字形相似度计算模块，保留了字形相似程度在 0.8 以上的字符，并过滤了生僻字。此外，我们也使用了一些前人开源的混淆集对上述各混淆集进行了扩充，最终所得混淆集对于资格集中别字错误的覆盖率达到 92.47%。

表 9 混淆集示例

混淆集名称	原字/词	混淆集内容
词级别同音混淆集	这时	这世|这是|这诗|着实|这使|这事|赭石...
字级别音近混淆集	妈	马|吗|码|麻|玛|嘛|骂|蚂|蛮|漫|满...
字级别形近混淆集	较	郊|咬|佼|饺|校|效|绞|蛟|交|胶|轿|皎...

除了直接生成数据外，我们还参考了英文领域的 Wiki-Edits 数据集[7]构建方式，利用中文维基百科编辑历史抽取并过滤得到了千万级别的中文 Wiki-Edits 平行语料。

4.2 通用标注数据

[1] https://github.com/nonamestreet/weixin_public_corpus
[2] https://github.com/brightmart/nlp_chinese_corpus
[3] http://101.200.120.155/

在本次评测的过程中，我们也搜集了以往的中文拼写纠错(Chinese Spelling Check, CSC)和中文语法纠错(Grammatical Error Correction, GEC)评测任务中常用的一些公开数据集。主要包含以下几部分：

(1) SIGHAN 拼写纠错数据[1]：2013 到 2015 年间，台湾大学等组织联合举办的 SIGHAN 拼写纠错评测任务公开的数据集，训练数据和测试数据合计约 1 万条。

(2) Lang-8 语法纠错数据[2]：根据 Lang-8 语言学习网站中母语者对汉语学习者作文的修改记录生成，约 122 万条平行数据。

(3) HSK 语法纠错数据[3]：北京语言大学对汉语学习者参加汉语水平考试中写的作文组织人员进行了语病标注，共约 15 万条平行数据。

5 实验

5.1 训练设置

拼写纠错模型结构如 3.1 节所述，主要使用 Pytorch 库搭建，其中 BERT 部分使用 transformers 库构建。具体的训练流程为：(1)使用 4.1 节中所述规则加噪方式中的替换规则构建规模约为 1.2 亿的伪数据进行预训练；(2)使用从中文 Wiki-Edits、HSK、Lang8 三份语法纠错数据中筛选出的约 600 万条拼写纠错数据进行模型的微调；(3)使用约 1 万条人工标注的 SIGHAN 拼写纠错数据进行模型的精调。

语法纠错模型结构如 3.2 节所述，核心代码和参数设置参考 GECToR 开源模型[4]。具体训练流程为：(1)首先对来自微信公众号语料库和中文新闻语料库的共 1500 万单语语料 A 使用拼写纠错模型进行推理，得到纠错结果 B；(2)然后对语料 A 使用 4.1.1 节中所述加噪规则进行加噪，得到人造语料 C；(3)最后将纠错结果 B 和人造语料 C 组成平行句对，用于语法纠错模型的训练。这样做可以防止原始语料本身存在的拼写错误对模型训练产生干扰，同时也可以将拼写纠错模型学习到的知识蒸馏一部分到语法纠错模型中。

此外，我们尝试使用 HSK 数据、HSK+Lang8 数据等人工标注的汉语学习者语法纠错数据集进一步微调语法纠错模型，并使用了基于规则和损失函数的数据清洗方法，但对系统性能都没有稳定的提升。我们猜测，正如本文第 2 节的数据分析所述，本次评测使用的测试数据与以往汉语学习者数据集存在着较大的领域差异，可能导致使用它们的效果不佳，因此最终我们所有的语法纠错模型都仅使用上述 1500 万人造数据进行训练。

5.2 资格集结果

拼写纠错方面，我们首先使用中文 BERT-WWM 预训练模型[8]初始化拼写纠错模型的编码器，然后按照 5.1 节中描述的训练设置，对模型进行了人造数据预训练(Pre-Train, PT)，并在 CTC-Qua 数据集上进行了测试(模型 CSC-PT)，可以看到虽然此时模型的召回度非常高，但精确度很低。而在引入了 3.1 节中的解码策略(Decoding Strategy, DS)后，可以大幅减少模型误纠的情况(模型 CSC-PT+DS)。此后我们又在真实数据上进行了多轮微调(Fine-Tune, FT)，进一步提升了模型的性能(模型 CSC-PT+DS+FT)。最后，我们使用阿里最新开源的中文 StructBERT 预训练模型[9]和哈工大开源的 MacBERT 预训练模型[8]重新初始化编码器，并且

---

[1] http://ir.itc.ntnu.edu.tw/lre/sighan8csc.html
[2] http://tcci.ccf.org.cn/conference/2018/taskdata.php
[3] https://github.com/shibing624/pycorrector#dataset
[4] https://github.com/grammarly/gector

更换了随机种子，训练了 2 个新的模型，并利用这 3 个模型进行集成(3.4 节描述方法)，得到了最优模型(模型 CSC-PT+DS+FT+3-Ens.)，可以在 CTC-Qua 数据集的别字类型达到约 70 的错误检测和错误纠正 F_1。

表 10 拼写纠错模型在 CTC-Qua 数据集别字错误上的结果

模型	别字错误检测			别字错误纠正		
	精确度	召回度	F_1 值	精确度	召回度	F_1 值
CSC-PT	43.11	**74.25**	54.55	41.04	**71.24**	52.08
CSC-PT+DS	51.34	70.57	59.44	49.76	68.9	57.78
CSC-PT+FT+DS	75.29	64.21	69.31	73.83	63.21	68.11
CSC-PT+FT+DS+3-Ens.	**81.3**	62.54	**70.7**	**80.43**	61.87	**69.94**

语法纠错方面，我们首先使用中文 MacBERT 模型初始化 GECToR 模型的编码器，然后采用 5.1 节中描述的训练设置进行训练，并在 CTC-Qua 数据集上进行测试(模型 GEC)。需要说明的是，表 11 最后一列展示的“得分”等于检测 F_1 乘以 0.8 加上纠错 F_1 乘以 0.2，是本次评测的最终排名依据，说明本次评测更看重模型的检错性能。为了充分利用模型集成技术带来的性能提升，我们又替换编码器为中文 StructBERT 和中文 RoBERTa 模型，更换了随机种子，并且重新生成了训练数据，训练了 2 个新的模型，共 3 个模型用于模型集成，获得了显著的性能提升(模型 GEC-3 Ens.)。接下来，我们又加入了表 10 中最优的拼写纠错模型，使用其在语法纠错前预先对数据进行拼写纠错，实验结果表明，拼写纠错模型可以和语法纠错模型互补，从而提升性能(模型 CSC+GEC-3 Ens.)。最后，我们又加入 3.3 节中的基于语义模板(Semantic Pattern, SP)的规则纠错方法(模型 CSC+GEC-3 Ens.+SP)，以及 3.4 节中的后处理(Post-Processing, PP)方法(模型 CSC+GEC-3 Ens.+SP+PP)，最终模型在 CTC-Qua 数据集上总得分为 62.8。

表 11 最终模型在 CTC-Qua 数据集的结果

模型	检错指标			纠错指标			得分
	精确度	召回度	F_1 值	精确度	召回度	F_1 值	
GEC	52.6	53.9	53.3	49.4	50.1	50.4	52.7
GEC-3 Ens.	55.9	57.5	56.7	52.3	53.9	53.1	55.7
CSC+GEC-3Ens.	54.4	61.3	57.7	51.1	58.8	54.7	57.1
CSC+GEC-3 Ens.+SP	55.8	**68.9**	61.7	52.9	**67.3**	59.2	61.2
CSC+GEC-3 Ens.+SP+PP	**60.4**	66.2	**63.1**	**57.1**	66.5	**61.4**	**62.8**

5.3 测试集结果

本次比赛的决赛阶段，每支队伍仅可提交 3 次系统，因此我们选择了 3 种差异较大的策略：第 1 次提交通过增加训练模型时的微调轮数，尽可能保证模型纠错的精度；第 2 次提交增加了伪数据的规模，并且加大集成模型的个数，尽可能提高纠错的召回度；第 3 次提交则进一步增加数据量和模型个数，引入了后处理干预规则，并且在模型推理时过滤掉置信度较低的纠正。表 12 展示了 3 次提交的系统性能。

表 12 在 CTC-Test 数据集上 3 次提交的结果

Runs	检错 F_1 值	纠错 F_1 值	得分
1	63.6	59.4	62.8
2	64.0	60.7	63.3
3	**68.0**	**64.6**	**67.3**

最终，我们提交的第 3 版系统在测试集上的检错和纠错性能位列第 1，最高得分为 67.3。表 13 展示了本次比赛排名前 5 的队伍指标，S&A 为本队队名。

表 13 排名前 5 的队伍指标

排名	队名	检错 F_1 值	纠错 F_1 值	得分
1	S&A	**68.0**	**64.6**	**67.3**
2	改的都队	62.4	57.2	61.4
3	znv_sentosa	55.0	43.1	52.6
4	C&L	51.1	48.6	50.6
5	MDatai	51.2	47.4	50.5

6 结语

在本次 CTC2021 评测任务中，我们使用了拼写纠错-语法纠错-语义模板纠错的 3 阶段纠错方案，并且提出了针对汉语母语者文本错误数据集稀缺问题的规则数据增强方法，还使用了一些额外的性能提升技术，例如基于混淆集的拼写纠错解码约束等。实验结果表明，我们提出的多种策略均可以使模型性能得到有效的提升，最终纠错系统在测试集上的总得分为 67.3，位列第 1。

但是，本次的系统依旧存在着许多不足。例如，我们通过引入规则模板的方式对语病错误进行纠正，其性能依赖模板质量，且可扩展性较差，未来可以尝试从模型结构层面将语义信息引入纠错模型。此外，我们的拼写纠错模型和语法纠错模型都是字级别的，未来可以研究利用汉语特有的词语信息和丰富的词义信息进一步提升纠错模型的性能。

参考文献

[1] Zhao Y, Jiang N, Sun W, et al. Overview of the NLPCC 2018 shared task: Grammatical error correction[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 439-445.

[2] Rao G, Yang E, Zhang B. Overview of NLPTEA-2020 Shared Task for Chinese Grammatical Error Diagnosis[C]//Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications. 2020: 25-35.

[3] Hinson C, Huang H H, Chen H H. Heterogeneous Recycle Generation for Chinese Grammatical Error Correction[C]//Proceedings of the 28th International Conference on Computational Linguistics. 2020: 2191-2201.

[4] Kenton J D M W C, Toutanova L K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]//Proceedings of NAACL-HLT. 2019: 4171-4186.

[5] Omelianchuk K, Atrasevych V, Chernodub A, et al. GECToR–Grammatical Error Correction: Tag, Not Rewrite[C]//Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2020: 163-170.

[6] 张仰森, 郑佳. 中文文本语义错误侦测方法研究[J]. 计算机学报, 2016, 39.

[7] Grundkiewicz R, Junczys-Dowmunt M. The Wiked error corpus: A corpus of corrective Wikipedia edits and its application to grammatical error correction[C]//International Conference on Natural Language Processing. Springer, Cham, 2014: 478-490.

[8] Cui Y, Che W, Liu T, et al. Revisiting Pre-Trained Models for Chinese Natural Language Processing[C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings. 2020: 657-668.

[9] Wang W, Bi B, Yan M, et al. StructBERT: Incorporating Language Structures into Pre-training for Deep Language Understanding[C]//International Conference on Learning Representations. 2019.