CCL2022-CLTC赛道三：基于大规模序列标记集成方法的文本纠错

孙邱杰*，梁景贵，李思
北京邮电大学　人工智能学院，北京　100876
(*通信作者电子邮箱sunqiujie0@bupt.edu.cn)

摘要

文本纠错任务的目的是纠正自然文本中存在的语法错误,目前基于序列标记的文本纠错方法被广泛应用。近年来，大规模的中文预训练模型在众多任务上取得了比较优异的性能，且不同的预训练模型在特定任务上存在各自的优势，然而，各个模型以及不同集成方法对基于序列标记文本纠错模型的影响尚不明确。针对上述问题，本文研究了大规模中文预训练模型对基于序列标记的文本纠错模型性能的影响，并探索了使用模型输出概率平均和使用输出编辑跨度的多数投票进行模型集成对性能的影响。在CCL2022-CLTC赛道三数据集上的实验结果表明了大规模中文预训练模型对基于序列标记的文本纠错模型性能有显著提升，且证明了使用编辑跨度的多数投票进行模型集成方法的有效性。本文最好的集成模型在CCL2022-CLTC赛道三第二阶段测试集上位于排行榜第三名。

关键词： 大尺寸预训练模型 ； 集成 ； 文本纠错 ； 序列标记

CCL2022-CLTC Track3:Ensembling of Large Sequence Taggers for Text Correction

SUN Qiujie*,LIANG Jinggui,LI Si
school of Artificial Intelligence,
Beijing University of Posts and Telecommunications,
Beijing 100876
(*sunqiujie0@bupt.edu.cn)

Abstract

The purpose of text correction task is to correct grammatical errors in natural texts, in which methods based on sequence tagging are widely used. Recently, pretrained Chinese Transformer-based encoders in large configurations have been proved to be effective in many tasks, and different Transformer-based encoders their own advantages on specific tasks. However,the impact of ensembling of sequence taggers based on pretrained Chinese Transformer-based encoders in large configurations for text correction have not been explored. Focused on above issues, the improvements to the text corrections sequence tagging with recent cutting-edge Transformer-based encoders in Large configurations are investigated. Experimental results on CCL2022-CLTC Track3 datasets demonstrate that pretrained Chinese Transformer-based encoders in large configurations bring about significant improvement to sequence taggers for text correction. And the effectiveness of ensembing models by majority votes on output edit spans have been proved. Our best ensemble ranks third in the leaderboard on CCL2022-CLTC Track3 phase 2 test set.

Keywords: Large pretrained Transformer-based encoders , Ensemble , Text correction , Sequence tagging

1 引言

文本纠错任务的目标是利用自然语言处理技术，自动识别并纠正自然语言文本中包含的拼写、标点符号、语法、词法、选词等方面的语法错误。同一个语法错误从不同语法点的角度可以被划定成不同的性质和类型 (张宝林, 2013)，也会因为语言使用的场景不同、具体需求不同，存在多种正确的修改方案。一个智能的文本纠错系统可以接收包含语法错误的文本并生成其对应的更正版本。文本纠错任务是一项复杂且具有挑战性的任务，其中编辑的准确性、推理速度和内存研究都是目前主要研究的主题。文本纠错任务可以支撑或应用于搜索 (Martins and Silva, 2004; Gao et al., 2010)、光学字符识别 (Afli et al., 2016; Wang et al., 2018)、文章评分 (Burstein and Chodorow, 1999)等领域。

目前，文本纠错任务常被当作机器翻译任务处理，将可能含有语法错误的句子作为源语句，不含语法错误的正确句子作为目标语句。近年来，基于Transformer的序列到序列模型(sequence to sequence, seq2seq)成为了文本纠错任务中的主流方法。该方法能自回归地捕捉输出序列中的全局依赖，但是由于模型是顺序解码，这使得模型推理速度比较慢。Grundkiewicz (2019)等人使用合成的文本纠错数据对Transformer模型进行预训练，并对集成结果从右到左进行重排序，取得了不错的结果。Kaneko (2020)等人提出了几种BERT的使用策略用于文本纠错。最近，Xue (2020)等人基于T5的seq2seq模型xxl版本(110亿参数)搭建了一个文本纠错系统，并取得了新的最佳结果(state-of-the-art, SOTA)。另外，基于序列标记的方法在文本纠错任务中逐渐被广泛应用，该类方法能有效缓解基于序列到序列模型推理速度缓慢的问题。Omelianchuk (2020)等人的GECToR系统加快了数倍模型推理速度，且取得了很有竞争力的性能。

在本文工作中，主要探索了基于GECToR (Tarnavskyi et al., 2022)的序列标记模型中使用不同的中文预训练模型以及不同集成方法和集成模型个数的效果。本文主要是通过使用不同的中文预训练模型作为GECToR系统中的Transformer编码器，并探索了使用模型输出标记概率平均进行集成和使用模型输出编辑跨度的多数投票进行集成的效果。本文的主要工作如下:

1. 本文通过将GECToR序列标记系统中的Transformer编码器更新为其他更大规模的中文预训练模型，通过实验证实了大规模的中文预训练模型的有效性，最好的单个模型(chinese-roberta-wwm-ext-large)并在CCL的YACLC赛道三第二阶段取得了39.57的F0.5值。

2. 本文通过实验表明，与使用模型输出标记概率平均来集成模型相比，通过输出编辑跨度的多数投票来集成模型提供了更好的性能。

2 背景

2.1 参赛任务

汉语学习者文本纠错任务(Chinese Learner Text Correction, CLTC)的目标是自动识别并修改汉语学习者文本中的标点、拼写、语法、语义等错误，从而获得符合原意的正确句子。随着该任务越来越受到关注，并出现一些具有潜在商业价值的应用，研究者旨在建立多参考答案的评价标准，完善文本纠错数据及任务，聚焦该研究领域中的前沿问题，进一步推动汉语学习者文本纠错研究的发展 (王莹莹et al., 2022)。

多维度汉语学习者文本纠错针对一个句子的多个参考答案，从最小改动(Minimal Edit, M)和流利提升(Fluency Edit, F)两个维度对模型结果进行评测。最小改动维度要求尽可能好地维持原句的结构，尽可能少地增删、替换句中的词语，使句子符合汉语语法规则；流利提升维度则进一步要求将句子修改得更为流利和地道，符合汉语母语者的表达习惯。表1给出了一个多维度多参考中文语法纠错任务的示例，其中输入序列为可能含有语法错误的原句，输出序列为对应的纠正语法错误后的句子，参考答案为原句对应的不包含语法错误的正确句子。

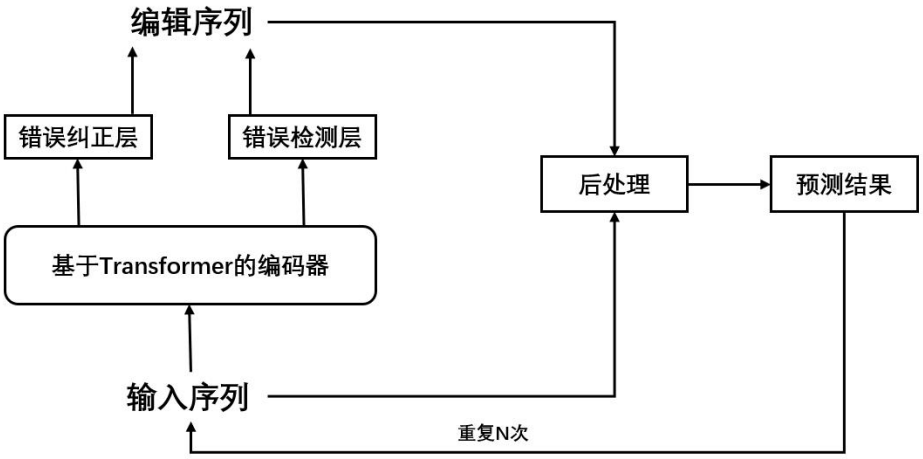

图 1: GECToR模型结构图

Table 1: 多维度多参考中文语法纠错任务示例

原句(输入序列)	因为我的中文没有好，我还要努力学汉语。	
最小改动	参考答案1	因为我的中文~~没有~~ 不好，我还要~~在~~努力学汉语。
	参考答案2	因为我的中文~~没有~~ 不好，所以我还要努力学汉语。
	纠正后句子(输出序列)	因为我的中文~~没有~~ 不好，我还要~~在~~努力学汉语。
流利提升	参考答案1	~~因为~~我的中文没有那么好，因此我还要努力学习汉语。
	参考答案2	因为我的中文还没有学好，所以我还要更加努力地学~~汉语~~中文。
	纠正后句子(输出序列)	因为我的中文没有那么好，所以我还要努力学汉语。

2.2 相关工作

文本纠错中的序列标记(sequence to edit, seq2edit)方法虽然还没有在机器翻译中普及，但它能为错误的输入文本生成由标记编码的文本编辑操作序列，这能有效加快模型的推理速度。Malmi (2019)等人提出的LaserTagger模型将文本生成视为一个文本编辑任务，使用保留标记、删除标记和在标记前添加一个短语三种主要的编辑操作将输入文本重新构建生成修正后的文本。LaserTagger结合了BERT编码器和一个自回归的Transformer解码器，并且可以预测编辑操作。Awasthi (2019)等人提出的并行迭代解码(Parallel Iterative Edit, PIE)模型能够进行并行解码，并且能达到与seq2seq模型相媲美的性能。Omelianchuk (2020)等人提出了一个类似的GECToR系统，使用Transformer作为一个编码器，后接一个线性层用于标记的预测和错误检测。通过将自回归的Transformer解码器替换成线性层，能使得模型推理速度比seq2seq系统快数倍，并且能取得很有竞争力的结果。

3 系统

3.1 GECToR模型结构

本文使用的基线标记模型是GECToR模型，它是基于AllenNLP和HuggingFace库完成并且实现了代码开源，该模型在英文语法纠错基准集CoNLL-2014和BEA-2019都取得了接近SOTA的效果。

GECToR模型结构如图1所示，GECToR是一个序列标记模型，主要由一个基于Transformer的编码器和两个输出线性层组成，其中两个线性层分别负责错误检测和错误纠正。该模型使用交叉熵损失函数进行训练，从而生成包含字符级编辑编码信息的标记，然后通过迭代后处理。GECToR可以预测输入序列中每个字符的标记，并将其转换成对应的编辑操作，再通过这些操作来获得修改后的输出序列。因为句子中的一些修正可能依赖于其他修正，仅仅使用一次GEC序列标记可能不足以完全修正句子。因此，GECToR使用迭代修正的方法，通过在模型中重复运行句子来修改句子。

Table 2: CCL2022-CLTC赛道三第二阶段排行榜前五队伍(测试集)

	Average	**Minimal**			**Fluency**		
	$F_{0.5}$	**$F_{0.5}$**	**Prec**	**Rec**	**$F_{0.5}$**	**Prec**	**Rec**
kk	**55.74**	**71.51**	**79.95**	**50.27**	**39.97**	**50.69**	21.66
改正带小助手	50.41	64.42	69.99	48.86	36.4	43.01	22.54
BUPTCL	47.39	59.89	68.88	39.35	34.89	45.96	17.77
NLP的未来	47.12	59.35	65.1	37.07	34.89	42.44	20.39
BERT 4EVER	45.84	58.83	68.96	37.07	32.84	44.87	15.85

Table 3: 不同单个预训练模型的结果(开发集)

Encoder	**Average**	**Minimal**			**Fluency**		
	$F_{0.5}$	**$F_{0.5}$**	**Prec**	**Rec**	**$F_{0.5}$**	**Prec**	**Rec**
chinese-bert-wwm-ext	39.76	54.05	61.06	37.04	25.47	33.91	12.76
chinese-macbert-base	39.05	53.25	60.19	36.45	24.84	32.19	12.98
chinese-macbert-large	40.03	54.38	61.33	37.41	25.68	34.16	12.88
rbt6	39.56	53.80	60.79	36.84	25.32	33.77	12.65
rbt13	39.77	54.10	61.16	37.01	25.43	33.92	12.71
chinese-roberta-wwm-ext	**41.16**	**56.07**	**63.47**	**38.23**	**26.25**	**34.79**	**13.25**
chinese-roberta-wwm-ext-large	40.82	55.69	63.08	37.92	25.59	34.43	13.07

在GECToR模型中，对源序列$(x_1 \dots x_n)$进行一个自定义的转换操作$T(x_i)$得到目标序列。对于有限的字典大小，这样的转换操作增加了针对最常见语法错误纠正的覆盖率。初步的编辑操作会被编码器编码为以下标记：\$KEEP(保持当前字x_i不变),\$DELETE(删除当前字x_i),\$$APPEND\_t_1$(将字$t_1$添加在当前字$x_i$后),\$$REPLACE_t_2$(将当前字$x_i$替换为字$t_2$)。

3.2 本文方法

分词器：在GECToR模型中，字节对编码器使用的是一个自定义版本，因为AllenNLP库中的分词器太慢，而HuggingFace Transformers库中的分词器没有提供一个字节对编码到单词的映射。本文工作完全是基于HuggingFace Transformers库中的Transformers模型完成。另外，本文使用了HuggingFace最近发布的快速分词器，本文模型的编码器使用的分词器与HuggingFace在预训练时使用的一样，这样使得模型微调后能取得更好的效果。

初始化和训练设置：编码器权重参数默认加载预训练模型的权重参数进行初始化，线性层进行随机初始化。模型训练使用默认超参数的Adam优化器 (Kingma and Adam, 2014)。损失函数使用多类分类交叉熵损失函数。另外，模型训练使用了早停法：模型的损失函数在验证集上有3个批次没有得到任何提升则会停止训练。其他训练设置是遵从CCL2022-CLTC赛道三中seq2edit的训练设置[1]。

更新编码器：CCL2022-CLTC赛道三中的seq2edit方法的GECToR基线模型使用的编码器是chinese-bert-wwm，本文尝试将编码器更新为chinese-roberta-wwm-ext、chinese-roberta-wwm-ext-large、chinese-macbert-base、chinese-macbert-large、chinese-bert-wwm-ext、rbt6、rbt13等，并且使用不同编码器的GECToR模型进行了实验复现。通常来说会选择base尺寸的预训练模型作为编码器，因为它们拥有更好的推理速度。并且通过实验发现chinese-roberta-wwm-ext会取得最好的效果。

集成：对于具有不同输出内容的模型集，集成是一种经过验证的提高模型质量的方法。最近的大多数语法纠错方法都是通过集成单个模型取得了SOTA结果。在本文工作中，探索了使用模型输出标记概率平均和使用输出编辑跨度的多数投票两种方法来集成模型的效果。对于N个训练好的语法纠错模型$(Model_1 \dots Model_N)$，对应的编码器为$(Encoder_1 \dots Encoder_N)$，对给定的输入序列$X = (x_1 \dots x_n)$。使用模型输出标记概率平

[1] https://github.com/blcuicall/CCL2022-CLTC/tree/main/baselines/track3/seq2edit

Table 4: 不同模型输出标记概率平均进行集成的结果(开发集)

Ensemble	Average	Minimal			Fluency		
	$F_{0.5}$	$F_{0.5}$	Prec	Rec	$F_{0.5}$	Prec	Rec
chinese-roberta-wwm-ext	41.16	56.07	63.47	**38.23**	26.25	34.79	**13.25**
chinese-roberta-wwm-ext-large	40.82	55.69	63.08	37.92	25.59	34.43	13.07
chinese-roberta-wwm-ext+chinese-roberta-wwm-ext-large	**44.29**	**61.68**	**80.28**	32.05	**26.89**	**46.24**	10.06
chinese-roberta-wwm-ext+chinese-bert-wwm-ext	44.46	61.57	78.37	33.14	27.34	44.61	10.73
chinese-roberta-wwm-ext+chinese-macbert-base	43.05	59.52	76.27	31.68	26.57	44.19	10.24
chinese-roberta-wwm-ext+rbt6	43.02	59.49	76.14	31.73	26.55	44.26	10.21
chinese-roberta-wwm-ext+chinese-bert-wwm-ext+chinese-macbert-base	45.07	62.76	81.92	32.43	27.37	46.75	10.30
chinese-roberta-wwm-ext+chinese-bert-wwm-ext+rbt6	45.06	62.66	82.19	31.98	27.56	46.71	10.44
chinese-roberta-wwm-ext+chinese-roberta-wwm-ext-large+chinese-bert-wwm-ext	**46.92**	**64.89**	**84.78**	**33.47**	**28.94**	**47.42**	**11.31**

均的方法是将输入序列X通过使用不同编码器编码的得到各个输出标记概率P_{Tags_i}，再将各个输出标记概率P_{Tags_i}进行概率平均得到最终输出标记概率P_{Tags_ens}，再通过后处理得到修正后句子$Corr_sents_ens$。

$$P_{Tags\_i} = Encoder_i(X) \tag{1}$$

$$P_{Tags\_ens} = \frac{\sum_{i=1}^{N} P_{Tags\_i}}{N} \tag{2}$$

$$Corr\_sents\_ens = Postprocess(P_{Tags\_ens}, X) \tag{3}$$

使用输出编辑跨度的多数投票是将输入序列X通过不同的语法纠错模型得到各个纠正后的句子$Corr_sents_i$，再对各个模型纠正后的句子中存在的编辑操作进行多数投票得到最终的修正后句子。

$$Corr\_sents\_i = Model_i(X) \tag{4}$$

$$Corr\_sents\_ens = Major\_Votes(Corr\_sents\_i) \tag{5}$$

4 实验

如表2所示，本文最好的集成模型在CCL2022-CLTC赛道三的第二阶段测试集上取得平均47.39的$F_{0.5}$值，分别在最小改动和流利提升维度上取得了59.89和34.89的$F_{0.5}$值，位于第二阶段测试排行榜第三名，BUPTCL为本队队名。

4.1 更新编码器

本文尝试将GECToR模型的编码器替换为更新或者更大规模的预训练模型，并进行了实验复现，表3给出了不同单个预训练模型作为GECToR编码器在CCL2022-CLTC赛道三开发集上的实验结果。我们发现使用chinese-roberta-wwm-ext作为编码器的单个模型取得了最好效果，更大参数尺寸的chinese-roberta-wwm-ext-large反而会导致模型性能降低，这可能是因为实验训练数据数量不足，导致了大规模的预训练模型会出现欠拟合的情况，从而导致性能的降低。

4.2 集成

表4给出了在CCL2022-CLTC赛道三开发集上使用不同模型输出标记概率平均进行集成的实验结果。本文发现通过使用不同模型输出标记概率平均进行集成都能普遍提升模型的纠错性能，并且发现一般来说集成的模型数量越多，取得的结果会越好。另外，实验表明使用同一个编码器模型结构的不同尺寸进行集成可以比各自单独集成取得更好的效果。

表5给出了在CCL2022-CLTC赛道三开发集上使用输出跨度编辑的多数投票进行集成($\oplus$)与使用输出标记概率平均进行集成(+)的比较结果。本文发现使用输出跨度编辑的多数投票进行集成比使用输出标记概率平均进行集成取得更好的性能，主要表现在前者能取得更好的精确度。其中，chinese-roberta-wwm-ext$\oplus$chinese-roberta-wwm-ext-large$\oplus$chinese-bert-wwm-ext在开发集上取得了最好的性能。

Table 5: 输出跨度编辑的多数投票集成与输出标记概率平均集成的对比结果(开发集)

Ensemble	Average	Minimal			Fluency		
	$F_{0.5}$	$F_{0.5}$	Prec	Rec	$F_{0.5}$	Prec	Rec
chinese-roberta-wwm-ext+chinese-bert-wwm-ext+chinese-macbert-base	45.07	62.76	81.92	32.43	27.37	46.75	10.30
chinese-roberta-wwm-ext⊕chinese-bert-wwm-ext⊕chinese-macbert-base	46.19	64.36	86.12	32.01	28.01	50.06	10.14
chinese-roberta-wwm-ext+chinese-bert-wwm-ext+rbt6	45.06	62.66	82.19	31.98	27.56	46.71	10.44
chinese-roberta-wwm-ex⊕chinese-bert-wwm-ext⊕rbt6	46.28	64.44	86.03	32.16	28.12	49.74	10.27
chinese-roberta-wwm-ext+chinese-roberta-wwm-ext-large+chinese-bert-wwm-ext	46.92	64.89	84.78	**33.47**	28.94	47.42	**11.31**
chinese-roberta-wwm-ext⊕chinese-roberta-wwm-ext-large⊕chinese-bert-wwm-ext	**47.75**	**66.17**	**88.29**	33.05	**29.33**	**51.2**	10.83

Table 6: 在多维度上本文模型和基线模型纠错对比示例

		最小改动	流利提升
示例1	输入	每年五月燕子来我住的地方，筑巢。	这个月我和老公去的餐厅的话，一个人2，000日元，而且很好吃。
	基线模型	每年五月燕子**都会**来我住的地方，筑巢。	这个月我和老公去的餐厅~~的话~~，一个人2000日元，而且很好吃。
	本文模型	每年五月燕子**都会**来我住的地方~~，~~筑巢。	这个月我和老公去的餐厅~~的话~~，一个人**花了**2000日元，而且很好吃。
示例2	输入	那地方不很方便，我们做打的去了那儿。	在那里我们看电影，喝可乐了。
	基线模型	那地方不**是**很方便，我们做打的去了那儿。	在那里我们看电影，喝可乐了。
	本文模型	那地方不**是**很方便，我们~~做~~打的去了那儿。	在那里我们看**了**电影，喝**了**可乐。
示例3	输入	我一定没有对他恋爱感情把！	所以大半的日本人不知道怎么很难学习外语。
	基线模型	我一定~~没有~~对他**有**恋~~爱~~感情把**吧**！	所以大半的日本人不知道~~怎么~~很难学习外语。
	本文模型	我一定没有对他**有**恋~~爱~~感情把**吧**！	所以大半的日本人不知道怎么~~很~~难学习外语。

4.3 示例分析

表6给出了在最小改动和流利提升两个维度上，本文模型和基线模型纠错的对比示例。由表格中示例结果可以看出，本文模型相比于基线模型，模型纠错时能覆盖到更多的错误。例如在最小改动维度的示例1中，本文模型与基线模型都对句子中缺失词“**都是**”进行了添加，但本文模型还删除了句子中的冗余字符“，”；示例2中本文模型也删除了冗余字符“**做**”，这是基线模型没有修正的错误。同样，在流利提升维度的示例1中，本文模型对缺失词“**花了**”进行了添加，示例2中添加了语气助词“**了**”，这些都是基线模型没有纠正的错误。另外，本文模型相比于基线模型，少了一些过度修改的错误。例如在最小改动维度的示例3中，基线模型对句子进行了过度修改，删除了原本正确的词“**没有**”，而本文模型则对其进行了保留；在流利提升维度的示例3中，本文模型也对原本正确的词“**怎么**”进行了保留，没有进行过度修改。

5 总结

本文主要探索了编码器配置和集成方法对GECToR模型的影响。本文通过实验发现使用更大规模的预训练模型作为编码器能够取得更好的性能，本文最好的单个模型chinese-roberta-wwm-ext在CCL2022-CLTC赛道三开发集上取得了41.6的$F_{0.5}$值。另外，我们发现使用输出跨度编辑的多数投票进行集成比使用输出标记概率平均进行集成能取得更好的性能，本文最好的集成模型chinese-roberta-wwm-ext⊕chinese-roberta-wwm-ext-large⊕chinese-bert-wwm-ext在开发集上取得了最好的性能，在CCL2022-CLTC赛道三第二阶段测试集上取得了47.39的$F_{0.5}$值，位于排行榜第三。另外，本文将代码进行了开源[1]。

实验结果表明，使用单个模型时，使用更大参数尺寸的chinese-roberta-wwm-ext-large会比使用chinese-roberta-wwm-ext作为编码器性能更低，这主要可能时因为训练数据数量不足，导致了大规模的预训练模型会出现欠拟合的情况，从而导致模型性能不佳。在未来工作中，数据增强方法将被考虑用于数据合成，并探索将合成数据加入大规模模型预训练后对性能的影响。

致谢

我们对北京邮电大学模式识别实验室李思导师长期以来的指导表示衷心的感谢，并且感谢梁景贵学长的指导帮助。另外我们对九天毕平台对实验室提供的算力帮助表示感谢。

参考文献

Haithem Afli, Zhengwei Qui, Andy Way, and Páraic Sheridan. 2016. Using smt for ocr error correction of historical texts.

[1] https://github.com/Anddyyyyy/CCL2022_TRACK3

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. arXiv preprint arXiv:1910.02893.

Jill Burstein and Martin Chodorow. 1999. Automated essay scoring for nonnative english speakers. In Computer mediated language assessment and evaluation in natural language processing.

Jianfeng Gao, Chris Quirk, et al. 2010. A large scale ranker-based system for search query spelling correction. In The 23rd International Conference on Computational Linguistics.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 252–263.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. arXiv preprint arXiv:2005.00987.

Diederik P Kingma and Jimmy Ba Adam. 2014. A method for stochastic optimization. arxiv e-prints, page. arXiv preprint arXiv:1412.6980, 5.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. arXiv preprint arXiv:1909.01187.

Bruno Martins and Mário J Silva. 2004. Spelling correction for search engine queries. In International Conference on Natural Language Processing (in Spain), pages 372–383. Springer.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: tag, not rewrite. arXiv preprint arXiv:2005.12592.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. arXiv preprint arXiv:2203.13064.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2517–2527.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. arXiv preprint arXiv:2010.11934.

张宝林. 2013. 关于通用型汉语中介语语料库标注模式的再认识. 世界汉语教学, 27(1):128–140.

王莹莹, 孔存良, 刘鑫, 方雪至, 章岳, 梁念宁, 周天硕, 廖田昕, 杨麟儿, 李正华, 饶高琦, 刘正皓, 李辰, 杨尔弘, 张民, and 孙茂松. 2022. Cltc 2022：汉语学习者文本纠错技术评测及研究综述.