论文题目:基于深度学习的中文文本自动校对研究与实现

学位类别：工学硕士

学科专业：软件工程

年　　级：2016

研 究 生：杨宗霖

指导教师：李天瑞

二零一九年五月

国内图书分类号：TP311　　密级：公开
国际图书分类号：004

西 南 交 通 大 学
研 究 生 学 位 论 文

基于深度学习的中文文本自动校对研究与实现

年　　级　2016
姓　　名　杨宗霖
申请学位级别　硕士
专　　业　软件工程
指 导 教 师　李天瑞

二零一九年五月十五日

Classified Index: TP311
U.D.C:004

Southwest Jiaotong University

Master Degree Thesis

Research and Implementation of Chinese Text Automatic Proofreading Based on Deep Learning

Grade: 2016
Candidate: Yang Zonglin
Academic Degree Applied for: Master Degree
Speciality: Software Engineering
Supervisor: Li Tianrui

May 15, 2019

西南交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1．保密□，在_____年解密后适用本授权书；

2．不保密☑，使用本授权书。

（请在以上方框内打“ √ ”）

学位论文作者签名：　　指导老师签名：

日期：2019 年　月　　日　　日期：2019 年　月　　日

西南交通大学硕士学位论文主要工作（贡献）声明

本人在学位论文中所做的主要工作或贡献如下：

（1）预处理 NLPCC 2018 语法错误纠正（Grammatical Error Correction，GEC）共享任务训练集，得到的平行语料用于训练中文校对模型。预处理维基百科中文语料，切分后的文本用于预训练词向量和统计 N-gram 语言模型。借助官方分词工具对校对系统输出进行重切分，基于官方性能评估脚本计算校对任务标准评估指标。

（2）提出一种基于字级别卷积编解码网络的中文校对模型，该模型通过建模字序列有效解决了标准分词算法与中文校对场景不匹配的问题，通过卷积操作使得能够更有效地利用局部上下文信息纠正语法错误。此外还通过数据扩增、数据清洗和预训练的字向量初始化嵌入矩阵进一步提升字级别中文校对模型的性能。

（3）提出一种基于集成解码和重排序的中文文本自动校对方法，该方法将重排序组件接在校对模型之后，针对 GEC 模型集成解码的 N 个最佳输出进行重打分，使得能够基于任务具体的特征和外部语言模型指导校对系统的纠正过程。重排序组件通过加权求和 GEC 模型解码器分数、针对校对任务的编辑操作计数特征和五元语言模型分数，得到输入错句对应的 N 个最佳候选输出新的分数，分数最高的句子即为输入错句的最佳纠正输出。

（4）提出一种基于多通道融合与重排序的中文文本自动校对方法。由于字级别的模型更擅长于校对错别字错误，子词级别的模型更擅长于校对真词搭配错误，多通道融合与重排序架构结合字级别和子词级别的校对模型，通过三个预测通道有效结合不同层次的信息，各预测通道均启用集成解码并输出 N 个最佳候选，最后再对聚合的多通道输出结果应用标准化的 LM 特征重排序打分并选出最佳输出。

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确说明。本人完全了解违反上述声明所引起的一切法律责任将由本人承担。

学位论文作者签名：

日期：

摘 要

互联网的高速发展使得广大用户能够生成海量的网络文本，对文本中隐含的语法错误进行校对可以使得行文更为流畅且易于阅读，而基于人工校对处理海量文本显然并不现实，这使得文本自动校对任务近年来受到较多关注。深度学习的兴起，使得序列到序列学习方法在文本校对任务中得到广泛应用，本文改进现有的基于深度学习的中文文本自动校对方法，主要从以下几个方面展开了工作：

（1）预处理训练数据和实现标准性能评估方法。预处理 NLPCC 2018 语法错误纠正（Grammatical Error Correction，GEC）共享任务训练集，将其转为校对平行语料，用于训练中文校对模型。预处理维基百科中文语料，切分后的文本用于训练词向量和统计语言模型。对校对系统输出的序列用官方分词工具进行重切分，再使用官方性能评估脚本计算校对任务标准评估指标。

（2）针对校对任务的性质，提出一种基于字级别卷积编解码网络的中文校对模型，并通过扩展训练平行语料、移除异常句子对和使用预训练的字向量初始化嵌入矩阵进一步提升模型性能。数据扩增对比实验表明，使用更多高质量的校对平行语料能显著增强模型性能。数据清洗对比实验表明，过滤目标端长度远小于源端的句子对能有效提升性能。不同级别建模对比实验表明字级别的切分粒度优于词级别和子词级别。嵌入矩阵不同方式初始化的对比实验表明，使用预训练的 word2vec 字向量能够有效提升字级别模型的性能。

（3）提出一种基于集成解码和重排序的中文文本自动校对方法，它针对 GEC 模型集成解码的 N 个最佳输出应用重排序机制，通过结合 GEC 模型解码器分数、针对校对任务的编辑操作特征和五元语言模型分数，对输入错句对应的 N 个最佳候选输出重打分，得分最高的句子即为输入错句的最佳纠正输出。相关实验结果表明，集成解码与重排序的组合是有效的，且重排序机制能显著提升模型性能。

（4）针对字级别和子词级别模型的局限性，提出一种基于多通道融合与重排序的中文文本自动校对方法。该方法通过三个预测通道结合字级别和子词级别的校对模型，各通道均启用集成解码机制并输出 N 个最佳候选，然后将各通道的输出结果聚合并应用标准化的语言模型特征重排序，分数最高的句子即为最佳输出。实验结果验证了该方法的有效性。

关键词：中文自动校对；神经机器翻译；卷积编解码网络；重排序；多通道融合

Abstract

The rapid development of the Internet enables a large number of users to generate a large amount of network text. Correcting the grammatical errors implied in the text can make the text smoother and easier to read. It is obviously unrealistic to process massive text based on manual proofreading, which makes the text automatic proofreading tasks have received much more attention in recent years. The rise of deep learning makes the sequence-to-sequence learning method widely used in text proofreading tasks. This thesis improves the existing Chinese text automatic proofreading method based on deep learning, mainly from the following aspects:

(1)Training data are pre-processed and standard performance evaluation methods are implemented. The NLPCC 2018 GEC shared task training set is preprocessed and converted into proofreading parallel corpus for training the Chinese proofreading model. The Wikipedia Chinese corpus is preprocessed, and the segmented text is used for pre-training word vectors and statistical language models. The sequence of the proofreading system output is re-segmented using the official word segmentation tool, and the official performance evaluation script is used to calculate the standard evaluation index.

(2) For the nature of proofreading tasks, a Chinese proofreading model based on char-level convolutional encoder-decoder network is proposed. The model performance is further improved by extending the training parallel corpus, removing the abnormal sentence pairs and initializing the embedding matrix using the pre-training word vector. Data fusion comparison experiments show that using more high-quality proofreading parallel corpus can significantly enhance the performance of the model. Data cleaning comparison experiments show that the filtering of sentence pairs which length of target side is much smaller than the source side can effectively improve the performance. Different levels of modeling comparison experiments show that char-level segmentation granularity is superior to word level and sub-word level. Comparison experiments of different initialization methods of the embedding matrix shows that using pre-trained word2vec char vectors can effectively improve the performance of char-level models.

(3) A Chinese text automatic proofreading method based on ensemble decoding and re-ranking is proposed. It applies re-ranking mechanism to the N-best output of GEC models ensemble decoding. The aforementioned re-ranking mechanism combines the GEC model decoder score, the edit operation features for the proofreading task, and the 5-gram language

model score to re-score the N-best candidate output corresponding to the input erroneous sentence, and the sentence with the highest score is the best corrected output of the input erroneous sentence. Experiments show that the combination of ensemble decoding and re-ranking is effective, and the re-ranking mechanism can significantly improve the performance of the model.

(4) Aiming at the limitations of char level and sub-word level models, a Chinese text automatic proofreading method based on multi-channel fusion and re-ranking is proposed. The method combines the char-level and sub-word-level proofreading models through three prediction channels, in which each channel enables the ensemble decoding mechanism and outputs N best candidates. Then it aggregates the output results of each channel and applies the standardized LM feature re-ranking mechanism. The sentence with the highest score is the best output. The experimental results show the effectiveness of the proposed method.

Key words: Chinese automatic proofreading; Neural machine translation; Convolutional encoder-decoder neural network; Re-ranking mechanism; Multi-channel fusion

目 录

第 1 章 绪 论 1
1.1 研究背景与意义 1
1.2 国内外研究现状 2
1.2.1 英文文本自动校对 2
1.2.2 中文文本自动校对 3
1.3 本论文研究内容及章节安排 4
1.3.1 本论文研究内容 4
1.3.2 本论文章节安排 5
1.4 本章小结 6
第 2 章 预备知识 7
2.1 卷积编解码网络 7
2.1.1 编解码网络 8
2.1.2 位置编码 8
2.1.3 多步注意力 9
2.2 解码生成方法 9
2.2.1 贪心解码 10
2.2.2 Beam search 解码 10
2.2.3 增量式解码 11
2.3 子词切分技术 12
2.4 词向量训练算法 13
2.4.1 word2vec 13
2.4.2 wang2vec 16
2.4.3 cw2vec 17
2.5 本章小结 18
第 3 章 实验设置与数据预处理 20
3.1 实验软硬件环境 20
3.1.1 硬件设备 20
3.1.2 软件环境 20
3.2 训练数据及其预处理 21
3.2.1 NLPCC 2018 GEC 训练集和数据处理 21
3.2.2 维基百科中文语料和数据处理 23

3.2.3 汉语水平考试平行语料 24
3.3 基准测试集和评价指标 25
3.3.1 基准测试集介绍 25
3.3.2 校对任务评价指标 25
3.4 本章小结 26
第 4 章 基于卷积编解码网络的中文校对模型 27
4.1 基于卷积编解码网络构建中文校对基线模型 27
4.1.1 基线模型的架构设置与训练细节 27
4.1.2 基线模型的性能表现 28
4.2 基于数据扩增与数据清洗提升中文校对模型的性能 28
4.2.1 数据扩增 28
4.2.2 数据清洗 29
4.3 基于字级别卷积编解码网络的中文校对模型 32
4.3.1 基于不同级别建模的中文文本自动校对 33
4.3.2 不同级别建模对比实验结果 33
4.4 基于预训练的词向量提升中文校对模型的性能 34
4.4.1 基于不同词向量算法训练不同级别的嵌入表示 34
4.4.2 基于不同的嵌入矩阵初始化方式的中文文本自动校对 35
4.4.3 嵌入矩阵不同初始化方式对比实验结果 36
4.5 本章小结 38
第 5 章 基于集成解码和重排序的中文文本自动校对 39
5.1 集成解码方法概述 39
5.2 基于重排序的中文文本自动校对 40
5.2.1 基于编辑操作特征的重排序 40
5.2.2 基于语言模型特征的重排序 43
5.2.3 基于编辑操作与语言模型特征的重排序 46
5.3 基于集成解码和重排序的中文文本自动校对 47
5.4 实验设计与结果分析 47
5.4.1 基于单模型推断和重排序的中文文本自动校对相关实验 47
5.4.2 基于集成解码和重排序的中文文本自动校对相关实验 48
5.4.3 不同中文校对系统在基准测试集上的性能对比 51
5.5 本章小结 52
第 6 章 基于多通道融合与重排序的中文文本自动校对 53
6.1 多通道融合与重排序架构 53

6.1.1 多通道预测组件 53
6.1.2 聚合组件 53
6.1.3 重排序组件 54
6.2 不同级别的校对模型 55
6.2.1 字级别的中文校对模型 55
6.2.2 子词级别的中文校对模型 55
6.3 实验设计与结果分析 56
6.3.1 不同中文校对系统的性能对比 56
6.3.2 不同重排序方法对比实验 57
6.3.3 组件拆解对比实验 58
6.4 本章小结 59
结论与展望 60
本文工作总结 60
未来工作展望 60
致 谢 62
参考文献 63
攻读硕士学位期间发表的论文及科研成果 69

第 1 章 绪 论

1.1 研究背景与意义

随着互联网与信息技术的高速发展，网络文本数量呈爆炸式增长，这对传统的以人工为主的校对提出了严峻挑战。为了降低人们的校对工作量，自动校对相关的研究工作得到了人们的重点关注。Chollampatt 和 Ng[1]指出英文语法错误纠正（Grammatical Error Correction，GEC）任务主要针对文本中存在的语法错误、拼写错误和搭配错误，Zheng 等[2]指出中文语法错误纠正（Chinese Grammatical Error Correction，CGEC）任务主要针对四种类型的语法错误：词语冗余（R）、词语缺漏（M）、搭配不当（S）和词序不当（W）。

早期的文本自动校对系统多基于规则、分类器和统计机器翻译（Statistical Machine Translation，SMT）。例如，文献[3][4]构建基于分类器的自动校对系统来解决英文文本校对的问题；文献[5][6][7]将文本校对视做转换错误文本为正确文本的单语翻译任务，将基于短语的统计机器翻译（Phrase-based SMT，PBMT）方法引入英文文本自动校对任务中；在中文校对领域，则主要有三类方法，即基于规则[8][9]，基于语言模型（Language Model，LM）[10][11][12][13]和基于知识库[14]。

近几年深度学习的兴起，使得一系列端到端的学习方法在自然语言处理（Natural Language Processing，NLP）领域得到应用，机器翻译（Machine Translation，MT）作为 NLP 领域的热点研究问题，一系列神经机器翻译（Neural Machine Translation，NMT）模型相继被提出，例如，RNN 序列到序列（sequence-to-sequence，seq2seq）模型[15][16]、注意力机制[17][18]、卷积 seq2seq（Convolutional Sequence to Sequence，ConvS2S）模型[19]和基于自注意力的 Transformer 模型[20]。

当前的 NMT 模型在上下文利用效率和输出序列流畅程度上都显著超越 SMT 系统，受此影响，当前主流的 GEC 方法也基本都基于 NMT 模型。例如，文献[21][22][23][1][24][25]充分利用机器翻译领域最近几年所取得的长足进展，构建了若干基于 NMT 模型的英文文本校对系统；文献[26][27][28]则基于 NMT 模型尝试解决中文自动校对问题。

三大主流的 NMT 模型在中文 GEC 领域相继得到应用。Zhou 等[26]使用 LSTM seq2seq 模型构建中文 GEC 系统，由于基于局部上下文信息即可纠正多数错误，且 LSTM[29]捕捉局部依赖的能力不如 CNN[30]，故 LSTM seq2seq 模型并不适合 GEC 任务。Fu 等[27]基于 Transformer 模型建模中文 GEC 任务，由于公开的中文校对平行语料仅为中等规模（100 多万平行句子对），而 Transformer 模型更为适合大规模的数据集（例如机器翻译的 WMT 数据集[20]），故 Transformer 模型不适合当前的 GEC 任务。Ren 等

[28]则基于 ConvS2S 模型构建了词级别和子词级别的中文校对模型，然而，他们忽略了字级别的建模，并使用 wang2vec[31]预训练嵌入表示增强模型性能。在中文 GEC 领域，标准的分词算法并不适合切分带有错误的文本，故本文提出基于字级别卷积编解码网络的中文校对模型，此外，本文还发现相较于 wang2vec，预训练的 word2vec[32][33]字向量更为有效，并通过数据扩增和数据清洗进一步提升字级别中文校对模型的性能。

重排序机制通过引入针对具体任务的特征和外部语言模型，对 MT 模型的 N 个最佳（n-best）输出进行重打分。Xie 等[22]使用 Common Crawl 语料的小子集训练 N 元（N-gram）语言模型做重排序。Ji 等[23]在重打分时使用了大规模的 Common Crawl N-gram 语言模型，但未使用任务具体的特征，即只使用了 NMT 解码器分数和语言模型分数对输出 n-best 句子重排序，且他们使用固定步长的网格搜索确定特征权重。Chollampatt 和 Ng[1]则在 Ji 等的基础上增加了编辑操作（Edit Operation，EO）特征，且使用最小错误率训练（Minimum Error Rate Training，MERT）[34]算法训练特征权重。本文在中文校对领域，针对 CGE 模型集成解码的 n-best 输出应用编辑操作与语言模型重排序，通过重打分机制指导校对过程，使得模型性能得到显著提升。

多模型融合方法在中文校对领域亦有得到应用。Zhou 等[26]通过层级结合的方式融合基于规则、基于 SMT 和基于 LSTM NMT 的组件，各组件均为单模型推断且只输出最佳句子，再通过语言模型从聚合的输出结果中选择最佳输出。Fu 等[27]则集成拼写错误纠正组件和多个不同配置和级别的 Transformer NMT 模型，各组件也均为单模型推断且只输出最佳候选，然后再对聚合的输出结果使用一个字级别的 5-gram 语言模型进行重打分。而本文则提出完全基于 ConvS2S 的多通道融合方法，通过三个预测通道结合字级别和子词级别的校对模型，其中各通道均启用集成解码机制并输出 N 个最佳候选，然后应用重排序组件基于标准化的语言模型特征对聚合的输出结果重打分。

1.2 国内外研究现状

1.2.1 英文文本自动校对

英文文本自动校对系统致力于纠正文本中的非词错误（拼写错误）、变形错误（时态和单复数等）、真词错误（词语搭配）和乱序错误（针对非原生语言使用者）等。CoNLL-2013[35]和 CoNLL-2014 GEC 共享任务[36]的成功举办使得文本校对任务得到了更多关注，一系列基于分类器和基于机器翻译的方法相继得到应用。

基于分类器的方法[4]需要针对不同类型的错误构建不同的分类器和混淆集，而基于 SMT 的方法则能从平行数据中自动学到混淆集，不需要其他的语言学输入，且使用一个 SMT 模型即能纠正多种错误类型，更擅长于校对复杂错误。尽管基于 SMT 的方法具有上述优点，但它依赖于大规模的人工标注平行语料，而基于分类器的方法则能够从无标注语料中学到模型。鉴于二者优缺点互补，2016 年 Rozovskaya 和 Roth[5]提出一

种管道架构，串联分类器和 SMT 系统，他们先应用分类器解决局部错误，再应用 SMT 系统解决复杂错误，这能有效结合分类器和 SMT 方法的优点。同年，Junczys-Dowmunt 和 Grundkiewicz[6]使用词级别的 SMT 模型和重打分机制取得了当时最先进的 GEC 性能，他们的重打分器使用了任务具体的特征（即目标句子和源句子间的编辑操作特征）和大规模的基于 Common Crawl 语料训练得到的 N-gram 语言模型。在此基础上，2017 年 Chollampatt 和 Ng[7]加入了自适应神经网络联合模型（NNJM）和基于字符级别 SMT 系统的拼写错误纠正器，在 CoNLL 2014 基准测试集[36]上取得了更好的性能。

然而，基于 SMT 的 GEC 系统受到泛化能力的限制，且不能够有效地访问更广泛的源端和目标端上下文，于是研究人员们将 NMT 方法应用到文本自动校对任务中，提出了一系列基于 RNN seq2seq 的模型。2016 年，Yuan 和 Briscoe[21]首次将经典的基于 RNN 和注意力的 NMT 模型[17]应用到 GEC 任务中，他们基于 CLC 语料[37]构建了词级别的模型，并使用无监督的词对齐模型和词级别的 SMT 模型来替换目标端输出的未知词。由于词级别的模型受到有限容量词汇表的限制，不能很好地处理未登录词，Xie 等[22]使用字符级别的编解码网络实现了开放词汇表。然而，完全基于字符的模型不能有效利用词级别的信息，于是 Ji 等[23]在 2017 年提出词字符混合校对模型，他们拓展了 Luong 和 Manning[38]的词字符混合机器翻译模型并加入了嵌套注意力层。

由于 RNN 是顺序编码序列的，并行程度不够，且捕捉局部上下文的能力不如 CNN，并且前述基于 RNN 的 GEC 方法[21][22][23]性能都没有超过最先进的（State-of-the-art，SOTA）基于 SMT 的系统。2018 年，Chollampatt 和 Ng[1]首次将完全基于 CNN 的 NMT 模型[19]应用到 GEC 任务中，他们使用预训练的 fasttext 词向量[39]初始化嵌入矩阵，基于 BPE[40]子词切分技术解决罕见词问题，通过应用 4 模型集成解码、重排序机制和拼写纠正器，使得该工作在众多神经方法中首次超过了 SOTA 的基于 SMT 的方法。由于数据增强有助于提升模型性能，且基于 NMT 的校对模型的输出序列可被反复编辑，Ge 等[24]提出了流畅度提升学习和推断机制，通过对偶提升学习和环形错误纠正技术，Ge 等在英文校对基准测试集上首次超越人类表现。由于公开可用的校对平行数据只有 Lang-8[41]和 NUCLE[42]，这导致公开的训练语料仅为中等规模，Lichtarge 等[25]使用比 Lang-8 大两个数量级的 Wikipedia 修订历史记录作为训练 GEC 模型的弱监督平行语料，然后基于维基数据集预训练 Transformer 模型，并将预训练的校对模型迁移到 Lang-8 语料上进行微调，最后使用迭代式解码完成推断过程。

1.2.2 中文文本自动校对

中文领域早期多用规则和统计学习方法解决校对任务。2002 年，徐连诚和石磊[8]使用基于动态规划的最长公共子序列长度求解算法，通过比对考生输入的有序字符集和标准答案，来解决计算机考试文字录入的自动阅卷问题。2003 年龚小谨等[9]基于两

阶段检查方法来检测中文文本的语法错误：第一遍全文扫描时，他们基于从错误语料中总结出来的搭配错误规则，通过模式匹配的方法检测搭配错误；第二遍全文扫描时，他们基于句型成分分析方法检查与句子成分相关的错误。同年，陈笑蓉等[10]结合二元词性模型、单字词共现概率极低的性质和相邻词的互信息对所构建的候选集进行排序，用第一候选字词纠正文本错误。

2016 年，刘亮亮和曹存根[11]基于 N-Gram 模型计算邻接二元概率和三元概率并加权求和，利用多特征融合模型的结果对混淆集进行排序，最后再通过规则标记错误真词的查错状态。刘亮亮和曹存根[12]还提出了一种模糊分词算法，旨在解决“非多字词错误”[43]。2017 年，张仰森和郑佳[14]通过词语搭配知识库和证据理论进行中文文本语义错误侦测。同年，张涛等[13]将校对转换成混淆集排歧的任务，他们通过正向最大策略进行组块，通过拼音相似和笔画相似获取当前组块的相似词列表，并借助字典树[44]加快词库的模糊搜索，最后基于二元模型打分排序并校对。

IJCNLP 2017 举办了中文语法错误诊断（Chinese Grammatical Error Diagnosis，CGED）共享任务[45]，该任务只要求检测出中文文本中的错误即可。Zheng 等[2]和 Xie 等[46]都将 CGED 任务视做序列标注问题，然后使用经典的 LSTM-CRF 模型求解。

NLPCC 2018 举行了首届语法错误校对共享任务测评[47]，吸引了多支队伍参赛。阿里巴巴的 NLP 团队[26]通过两阶段结合的方式融合基于规则、基于 SMT 和基于 NMT 的组件。网易有道 NLP 团队[27]则集成拼写错误纠正组件和多个不同配置和级别的 Transformer NMT 模型，并对聚合的输出结果使用一个字级别的 5-gram 语言模型进行重打分。北京语言大学（BLCU）的团队[28]则基于 BPE 级别的 ConvS2S 模型和 4 模型集成解码构建中文 GEC 系统。

1.3 本论文研究内容及章节安排

1.3.1 本论文研究内容

本学位论文主要的研究内容如下所示：

1. 训练数据的预处理和标准性能评估指标的实现方法

预处理 NLPCC 2018 GEC 共享任务训练集，得到的平行语料用于训练中文校对模型。预处理维基百科中文语料，切分后的文本用于训练词向量和统计 N-gram 语言模型。对校对系统输出的序列用官方分词工具进行重切分，再使用官方性能评估脚本计算校对任务标准评估指标。

2. 基于字级别卷积编解码网络的中文校对模型

首先基于 ConvS2S 构建本文的基线校对模型，随后基于数据扩增与数据清洗提升中文校对模型的性能，接着探讨了适用于中文校对任务的建模级别，并提出一种基于字级别卷积编解码网络的中文校对模型，最后使用预训练的 word2vec 嵌入表示增强校

对模型的性能。

3. 基于集成解码和重排序的中文文本自动校对

对中文校对模型集成解码的 n-best 输出应用重排序能显著提升模型性能，重打分机制能有效指导校对过程。本文的重排序组件支持基于编辑操作特征的重排序、基于语言模型特征的重排序和同时应用编辑操作与语言模型特征的重排序，通过训练脚本得到不同重排序机制所需的特征权重，然后在应用模型时指定要启用的重排序机制即可。对比实验表明，集成解码与重排序的组合是有必要的，且三种重排序方法均能显著提升模型性能。

4. 基于多通道融合与重排序的中文文本自动校对

多通道融合与重排序架构能够结合不同切分粒度的校对模型，能够利用不同级别的信息。该方法通过多个预测通道有效结合不同层次的信息，通过标准化的 LM 特征重排序组件对聚合的多通道输出结果进行重打分。相关对比实验表明了基于多通道融合与重排序的中文校对方法的有效性，组件拆解实验表明各组件都是有必要的。

1.3.2 本论文章节安排

全文共分 6 章。

第 1 章 绪论。介绍了论文的研究背景与意义、文本自动校对任务的国内外研究现状以及本论文的研究内容与章节安排。

第 2 章 预备知识。概述了相关背景知识，包括 ConvS2S 模型、seq2seq 模型的解码生成方法、子词切分技术（BPE）和三种词向量训练算法（word2vec、wang2vec 和 cw2vec）。

第 3 章 实验设置与数据预处理。首先介绍了本文实验的软硬件环境，包括使用的 GPU 设备和 fairseq NLP 库；然后介绍了三种训练数据（NLPCC 2018 GEC 训练数据、维基百科中文语料和 HSK 平行语料）并分别给出了它们的预处理方案；最后阐述了本实验所用的基准测试集和标准性能评估指标，并给出性能指标的计算流程。

第 4 章 基于卷积编解码网络的中文校对模型。首先确定了基线模型，然后基于数据扩增与数据清洗提升性能，接着提出一种基于字级别卷积编解码网络的中文校对模型，最后基于预训练的嵌入表示提升模型性能。

第 5 章 基于集成解码和重排序的中文文本自动校对。首先阐述了集成解码方法，随后提出三种针对中文校对模型 n-best 输出应用重排序的方法，接着提出基于集成解码和重排序的中文文本自动校对方法，最后是相关的实验设计与结果分析。

第 6 章 基于多通道融合与重排序的中文文本自动校对。首先阐述了多通道融合与重排序架构，接着确定了不同级别的校对模型，最后是相关对比实验分析。

1.4 本章小结

本章首先介绍了论文的研究背景与意义，阐述了本文工作的动机，接着就文本自动校对任务的国内外研究现状进行了介绍，最后概述了本论文的研究内容与章节安排。

第 2 章 预备知识

本章将概述与中文校对任务和本文实验相关的一些背景知识，包括卷积编解码网络、seq2seq 模型的解码生成方法（贪心解码、beam search 解码和增量式解码）、子词建模技术（BPE）和三种词向量训练算法（word2vec、wang2vec 和 cw2vec）。

2.1 卷积编解码网络

早期的校对任务大都基于规则、分类器和 SMT，当前主流的 GEC 方法则基本基于 NMT 模型，本文基于 ConvS2S 构建校对系统，故本小节将介绍卷积编解码网络。

RNN 相邻时间步间存在依赖关系，并行程度不够，为了能够充分利用 GPU 算力，Gehring 等[19]提出了完全基于卷积网络的 seq2seq 模型，ConvS2S 模型由编码器和解码器组成，如图 2-1 所示。

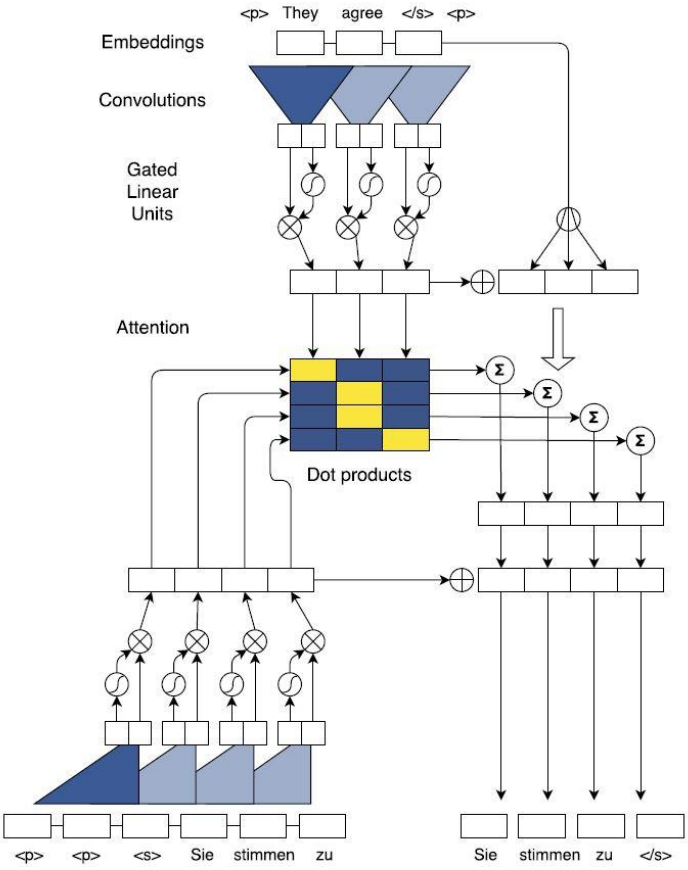

图 2-1 ConvS2S 模型架构图[19]

2.1.1 编解码网络

2.1.1.1 编码器

ConvS2S 模型编码器由若干编码层堆叠而成，每个编码层由 Conv1D 层（即一维卷积）、GLU 激活函数[51]和残差连接[52]三个组件构成。第一个编码层的输入为源端序列的嵌入表示，该嵌入表示为词向量与位置编码（见 2.1.2 小节）的求和。由于输入嵌入向量的维数与卷积核高度并不匹配，故输入编码层前需做一次线性映射。各卷积层输入和输出序列的长度一致，故需在卷积层输入的两侧进行零向量填充。

由于卷积过滤器的宽度取值一般较小，故 ConvS2S 编码器通过堆叠编码层的方式来增大感知野。各编码层输入和输出张量的形状完全一致，为了使得编码器输出向量的维数能够匹配多步注意力（见 2.1.3 小节）组件，编码器输出也需进行线性映射，使得维数变回输入嵌入向量的大小。

2.1.1.2 解码器

ConvS2S 模型解码器由若干解码层堆叠而成，每个解码层由 Conv1D 层、GLU 激活函数、多步注意力组件和残差连接组成。与 Stacked RNNs seq2seq NMT 模型[53]不同，ConvS2S 的每个解码层都配备了一个独立的编解码点积注意力组件，而 RNN NMT 模型一般只用第一层解码 RNN 的隐藏状态序列去和编码器的输出计算各解码时间步对应的语义编码向量c_t。

ConvS2S 模型在应用的时候也是通过自回归的方式顺序生成目标序列，ConvS2S 解码器的输入为之前预测输出的序列，通过对词向量和位置编码求和得到解码器的输入嵌入序列。对解码器输入序列应用线性映射，使得输入向量维数与卷积核的高度相匹配。各解码卷积层输入和输出序列长度一致，且解码器的隐藏状态不能读入来自后文的信息，故需在解码卷积层输入的左侧填充$k-1$个零向量，k 为卷积核的宽度。

解码层在计算注意力分数前需对解码层状态进行线性变换，使得维数与编码器输出相匹配，使得可以进行多步注意力的计算（见 2.1.3 小节）。多步注意力组件输出注意力上下文向量c_t，将c_t与解码层状态求和，再通过残差连接就得到了解码层的输出。解码器的输出也需要进行线性变换，使得维数变回输入嵌入的大小，最后通过 softmax 输出层预测下一个 token。

2.1.2 位置编码

RNN 网络由于顺序编码序列，其隐藏状态向量已建模各个词的位置信息。一维卷积将卷积窗口内的多个向量加权求和成一个元素，故处在同一卷积窗口内的多个词的处理方式类似于词袋模型，故通过编码 token 的绝对位置，将位置 id 转为向量表示，

使得能够建模位置信息，使得模型有感知位置的能力。

ConvS2S 将词 id 序列转为词向量序列$\mathbf{w}=(w_1,...,w_m)$，将位置 id 序列转为位置向量序列$\mathbf{p}=(p_1,.p_m$，再将词嵌入与位置编码求和，得到输入表示序列$\mathbf{s}=(w_1+p_1,...,w_m+p_m)$，再将$\mathbf{s}$馈入 ConvS2S 的编码器。解码器的输入也包含位置编码。位置向量嵌入矩阵随机初始化，与 ConvS2S 模型其他参数一起被训练。

2.1.3 多步注意力

ConvS2S 模型使用多步注意力组件计算目标时间步 t 与源端的对齐概率分布，然后输出各解码目标时间步对应的源端注意力语义编码c_t。ConvS2S 模型的多步注意力组件计算流程如下：

（1）将第 l 层第 n 个时间步维数为 h（h 为编解码卷积构建块输入和输出的维数）的解码层状态$y_n^l \in \mathbb{R}^h$依据式（2-1）线性映射为 d 维（d 为编解码器输入和输出向量的维数）的向量z_n^l：

$$z_n^l = \mathbf{W}_z y_n^l + b_z + t_{n-1} \tag{2-1}$$

其中，$\mathbf{W}_z \in \mathbb{R}^{d\times h}$为全连接层权值矩阵；$b_z \in \mathbb{R}^d$为偏置向量；$t_{n-1}$为上一个时间步预测输出的嵌入表示。解码层在计算注意力分数前需先对解码层状态y_n^l按式（2-1）进行线性变换，使得维数与编码器输出$\mathbf{e}=(e_1,...,e_m)$相匹配。

（2）将z_n^l与编码器输出$\mathbf{e}$计算点积注意力，通过 softmax 归一化得到目标端时间步 n 对源端的对齐概率$\alpha_{n,i}^l$，即注意力权重，$\alpha_{n,i}^l$由式（2-2）计算得到：

$$\alpha_{n,i}^l = \frac{\exp(e_i^{\mathrm{T}} z_n^l)}{\sum_{k=1}^{m}\exp(e_k^{\mathrm{T}} z_n^l)} \tag{2-2}$$

其中，T 代表转置符号。

（3）对编码器输出$\mathbf{e}$和源端输入$\mathbf{s}$的求和结果应用注意力权重$\alpha_{n,i}^l$得到注意力加权的源端表示x_n^l，这种$K \neq V$的注意力用法有助于注意力语义向量更好地保留源端输入的信息，x_n^l由式（2-3）计算得到：

$$x_n^l = \sum_{i=1}^{m}\alpha_{n,i}^l (e_i + s_i) \tag{2-3}$$

（4）x_n^l通过线性映射将维数从 d 变为 h，得到$c_n^l \in \mathbb{R}^h$，多步注意力组件输出上下文向量c_t即完成计算流程。

2.2 解码生成方法

当前主流 NMT 模型的解码方法基本基于自回归生成，近年也出现了一些非自回归解码的工作，例如 2018 年 Gu 等[54]基于繁殖力（fertility）预测机制将源端输入拷贝到目标端使得能够并行生成目标序列，2019 年 Guo 等[55]通过增强解码器的输入来减少翻

译精度的损失，Wang 等[56]通过相似度正则和重建约束来改善重复翻译和漏翻译的问题。相较于自回归解码的顺序生成，非自回归能并行预测目标端各时间步的输出，拥有更快的解码速度，但会导致翻译性能损失。当前 SOTA 的 NMT 模型为 Transformer，故当前的非自回归实现亦基于 Transformer 模型，而本文基于 ConvS2S 模型构建中文校对系统，故本文依然使用自回归解码。常见的自回归方法包括贪心解码（2.2.1 小节）、beam search 解码（2.2.2 小节）和增量式解码（2.2.3 小节）。

2.2.1 贪心解码

贪心解码每个解码时间步的预测都从输出分布中选择概率最大的 token，再将之前的输出全部回馈入解码器，通过自回归的方式顺序生成输出序列。贪心解码尽管简单直观，但无法得到解码推断的全局最优解。

2.2.2 Beam search 解码

Beam search 解码[57]通过引入集束宽度（beam width）超参 n，使得各解码时间步能保持当前 n 个最佳序列，最终时刻的 n-best 列表得分最高的序列即为模型推断输出，beam search 的解码流程如算法 1 所示。

算法 1 beam search 解码

Inputs：

n	集束宽度
nmt_model	训练好的 NMT 模型
source_sen	源端输入句子

全局变量：

results	列表，存储 n-best 序列
scores	列表，存储 n-best 序列的对数概率累加值

Output：

output_sen	目标端输出句子

1. results 初始化为[[<bos>的 id]]；
2. scores 初始化为[0]；
3. 将 source_sen 馈入 nmt_model 的编码器；//编码器只需计算一次
4. **while not** all seq∈results ends with <eos> **do**
5. 构造 tmp_results 空列表；
6. 构造 tmp_scores 空列表；
7. **for** seq **in** results **do**

8. **if** seq ends with <eos> **do**
9. 跳过这个序列；
10. **end if**
11. 将 seq 馈入 nmt_model 的解码器；
12. 对预测下一个 token 的概率分布求 log；
13. **for** index **in** range(词汇表的规模) **do**
14. 将 index 处的 log 概率和当前 seq 对应的 score 求和，并加入 tmp_scores；
15. 将 index 与 seq 合并，并加入 tmp_results；
16. **end for**
17. **end for**
18. results ← 合并 results 与 tmp_results；
19. scores ← 合并 scores 与 tmp_scores；
20. 按 seq 长度对 score 标准化并保留 top-n 序列；
21. 移除仅包含<bos>的序列；
22. 更新 results；
23. 更新 scores；
24. **end while**

//最后时间步的 n-best 列表得分最高的序列即为模型推断输出

25. 依据 scores 从 results 中选择 output_sen；

集束搜索本质也是一种贪心算法，设置合理的集束宽度能在裁剪的解空间中找到解码推断的较优解；若设置集束宽度为 1，则 beam search 解码退化为贪心解码；若设置集束宽度为词汇表规模，则计算量较大，解码效率降低。

2.2.3 增量式解码

自回归生成时需要不断把之前预测输出的 token 序列再回馈给解码器，每个解码时间步做预测时都需要重算整个解码器的状态序列，故可以用增量式解码解决重复计算的问题。

增量式解码通过缓存之前的隐藏状态使得后续推断不再需要重复计算，增量式解码的具体步骤为：

（1）每个解码时间步做预测时仅输入前一个时刻输出的向量表示。

（2）从缓存中恢复之前的隐藏状态。

（3）解码器做一步计算。

（4）使用最新状态更新缓存。

（5）基于当前时间步的解码隐藏状态去预测下一个 token。

2.3 子词切分技术

文本校对任务从根本上说是一个开放词汇表的问题，因为错误文本可能包含各种实体，例如人名、数字和日期等。当前主流的 GEC 方法基本都基于 NMT 模型，为缓解 softmax 函数计算量大的问题，词级别的基于 NMT 模型的 GEC 系统需要使用有限容量的词汇表。然而，有限容量的词汇表会导致罕见词变成集外词（OOV token），实践应用中一般会将 OOV token 替换成<UNK>特殊符号，而输入输出句子对中带有<UNK>符号会导致模型性能损失。

目前主要有两大类方法可以解决 OOV 词翻译的问题（本文将 GEC 任务视做翻译错误文本为正确文本的问题）：

（1）基于拷贝的方法，通过外部对齐模型或注意力对齐概率分布，将源端词直接拷贝到目标端或基于字典将词翻译为目标语言，以替换目标端输出的未知词。

（2）基于子词单元的方法，包括构建字符级别的模型、构建词字符混合的模型[38]和构建子词级别（BPE[40]或 WPM[58]）的模型。

本文使用 BPE（Byte Pair Encoding）编码解决词级别校对模型受限于罕见词的问题，具体方法为：

（1）设置合理的 BPE 操作次数（此为 BPE 算法唯一的超参），控制 BPE 词汇表（模型）的规模，经典的取值为 30000。

（2）基于校对平行语料的目标端通过 BPE 学习算法得到 BPE 词汇表。

（3）对校对平行语料应用 BPE 子词切分，将词序列转为 BPE 序列，训练 BPE 级别的校对模型。

（4）应用的时候，将 BPE 级别模型输出的 BPE 序列恢复为词序列。

BPE 基于训练语料学到词的子词切分方式，通过保留高频词和将低频词切分为高频子串单元序列，有效压缩了 NMT 模型源端和目标端词汇表的规模，BPE 学习算法如下所示：

（1）对语料进行词频统计。

（2）将词按照 BPE 模型进行最大匹配切分，得到 BPE token 和字符 token 的混合序列。

（3）统计 token 两两共现的频数，拼接共现次数最多的两个 token 并加入 BPE 词汇表中。

（4）BPE 操作次数减一，若还剩下 BPE 操作，则重复执行。

BPE 编码依据 BPE 模型对词进行最大匹配切分，将词序列转为 BPE 序列，子词切分前后对比如表 2-1 所示。被切开的词使用特殊的“@@”符号标记，故可以使用“sed s/@@ //g”命令（sed 为 Unix 的流编辑器）将 BPE 序列恢复为原始序列。

表 2-1 词切分和 BPE 切分对比

jieba 分词工具切分结果	BPE 子词切分结果
这个/ 游戏/ 叫龙/ 和/ 地下城/ 。	这个/ 游戏/ 叫@@/ 龙/ 和/ 地下@@/ 城/ 。

本文通过 subword-nmt 工具包（https://github.com/rsennrich/subword-nmt）将 BPE 机制应用到中文文本校对任务中，基于 learn_bpe.py 学习 BPE 模型，基于 apply_bpe.py 应用 BPE 子词切分。

2.4 词向量训练算法

使用预训练的词向量应当能增强词级别模型的性能。Chollampatt 和 Ng[1]在英文 GEC 任务中使用预训练的 BPE fasttext 词向量初始化嵌入矩阵，Ren 等[28]则在中文 GEC 任务上使用预训练的 wang2vec 词表征初始化 BPE 级别模型的嵌入矩阵，这两项工作表明预训练嵌入表示的使用能够提升 BPE 级别 GEC 模型的性能。于是，本文基于中文维基语料（预处理方法见 3.2.2 小节）预训练 token 的 word2vec（词向量基线模型，见 2.4.1 小节）、wang2vec（针对语法问题的词向量，见 2.4.2 小节）和 cw2vec（笔画信息增强的词向量，见 2.4.3 小节）向量表示。本小节将介绍前述三种词向量训练算法。

2.4.1 word2vec

Word2vec 是词表示学习的基线模型，它包括 CBOW（Continuous Bag-of-Words Model）和 skip-gram 两种模型[32]，使用层次 softmax 和负采样两种 softmax 函数近似计算方法[33]。

2.4.1.1 CBOW 和 skip-gram 模型

1954 年，语言学家 Harris 提出经典的“分布式假设”[59]：语义/词义相似的单词往往会出现在相似的上下文中。这一假设奠定了后续各种词向量的语言学基础。早在 2003 年，Bengio 等[60]便提出了基于神经网络的语言模型（NNLM），他们使用前馈全连接神经网络构建 N-gram LM，在优化语言模型的同时顺便学到词的嵌入表示。由于训练 NNLM 的开销较大，Mikolov 等[32]于 2013 年提出了 CBOW 和 skip-gram 模型，这两个模型省去了隐藏层，计算量大幅下降，使得可以在大规模语料上训练词表示，是专门用于学习词向量的模型。

CBOW 模型，即连续词袋模型，它的输入为滑动窗口上下文词的索引，通过查表操作从词嵌入矩阵中得到上下文词的向量表示，投影层将上下文的词向量求平均（投影层没有需要训练的参数，由于窗口中词的位置并不影响投影操作的结果，故其被称之为词袋模型），再将得到的局部序列表示馈送给 softmax 输出层，预测位于窗口中间

的词，故 CBOW 模型本质上就是一个 softmax 回归，就是一个感知机。窗口在训练语料上的滑动会生成一系列的输入输出对，这些训练样本为 CBOW 模型提供监督信号，嵌入矩阵在误差反向传播时随模型其他参数一起被更新。

Skip-gram 模型的输入为滑动窗口中间词的 id，经过词嵌入层与 softmax 输出层预测上下文中的另外一个词，是一个单输入单输出的模型。窗口滑动一次即可生成 $R\times 2$ 个训练样本，其中 R 为 $[1,C]$ 区间内的随机整数值，C 为中间词到窗口边缘的距离（即最大预测跨度），这些输入输出对用于训练 skip-gram 模型，嵌入矩阵与模型其他参数一起被更新。

CBOW 和 skip-gram 在更新嵌入矩阵时拥有不同的行为模式：

（1）CBOW 模型在反向传播时窗口上下文的词拥有相同的梯度，即预测中间词的误差信号以相同的梯度更新上下文词的嵌入表示。

（2）sikp-gram 模型在反向传播时预测上下文不同词的误差信号将以不同的梯度更新中间词的嵌入表示。

综上可得到如下的一些结论：

（1）CBOW 模型由于一个窗口只构造一个输入输出对，故其拥有更快的训练速度，然而它以相同的梯度更新上下文的词嵌入，故 CBOW 模型无法很好地学到罕见词的向量表示。

（2）skip-gram 模型由于一个窗口即可构造 $R\times 2$ 训练样本，故其较 CBOW 模型更慢收敛，然而它能较好地学到罕见词的嵌入表示。

实践应用中，一般选用 skip-gram 模型。

2.4.1.2 层次 softmax 和负采样

标准的 CBOW 和 skip-gram 模型本质都是 softmax 回归，而 softmax 函数计算量较大，故一般采用层次 softmax 和负采样做近似计算。

层次 softmax 依据词频构建哈夫曼树，预测时从根节点计算到叶子节点，时间复杂度为对数级别。

负采样基于词频分布从训练语料中采样得到负样本（超过阈值的高频词需进行下采样，否则低频词的词向量更新次数太少，可能会欠拟合），此处仅介绍对 skip-gram 模型应用负采样机制。负样本与当前窗口的中间词不在同一个上下文，正样本则为窗口中的其他词，它与中间词处在同一个上下文，于是多分类问题转为二分类问题，只需计算 sigmoid 函数即可。基于负采样的 skip-gram 模型数据流为：

（1）输入为中间词与正样本或负样本的词 id。

（2）通过嵌入层将词 id 转为词向量。

（3）中间词与正样本或负样本计算相似程度（一般为点积计算）。

（4）将相似度函数结果馈入 sigmoid 输出层。

由于层次 softmax 需要经过较多步才能预测罕见词，故实践应用中多使用负采样近似计算 softmax。

2.4.1.3 word2vec 官方训练工具包

本文使用官方的 word2vec 工具包（https://code.google.com/archive/p/word2vec/），相关参数的说明见表 2-2，包括参数名、意义、默认值和其他信息。

表 2-2 word2vec 官方工具包参数说明

参数名	意义	默认值	其他信息
-train	切分好的训练语料	无	无
-output	预训练词向量的保存路径	无	标准的词向量保存格式为：第一行是词汇表规模和向量维数，空格分隔；接下来的每一行是词和它对应的嵌入表示，空格分隔；向量中各维的浮点数也是空格分隔
-size	词向量的维数	100	按照语料规模与下游模型架构来设置
-window	设置最大的预测跨度	5	官方推荐 CBOW 设置为 5，skip-gram 设置为 10
-sample	设置词频阈值，出现频率大于阈值的词，负采样时会被随机下采样	1e-3	有用的取值范围：0~1e-5
-hs	是否使用层次 softmax 近似计算 softmax 函数	0（即不使用）	层次 softmax 能更好地学到罕见词的嵌入表示
-negative	负样本的个数	5	通常的取值范围为 3-10，0 即不使用
-threads	多线程并行加速词向量的训练过程，指定使用线程的个数	12	根据机器可用计算资源进行设置
-iter	至多训练多少轮迭代	5	无
-min-count	指定过滤阈值，构建词汇表时会过滤出现次数低于阈值的词	5	无

续表 2-2

-alpha	设置初始学习率	skip-gram 的初始值为 0.025，CBOW 的初始值为 0.05	无
-binary	是否以二进制形式保存结果向量	0（以 txt 格式保存）	0 即以标准词向量格式进行保存
-cbow	指定词向量学习算法	1	1 即使用 CBOW 模型，0 即 skip-gram

2.4.2 wang2vec

2.4.2.1 wang2vec 词向量学习算法

word2vec 是将词从 one-hot 编码嵌入到语义空间，是在学习词的语义表示，故 CBOW 和 skip-gram 模型在建模上下文词和中间词的相似程度时都是词序无关的（CBOW 基于词袋模型，而 skip-gram 用于预测上下文的输出层在窗口的各个位置均相同）。然而对于语法问题，词序很重要，于是 wang2vec[31]对 word2vec 进行了拓展，提出了结构化 skipngram 和 CWINDOW 两种模型，它们分别是标准 skip-gram 和 CBOW 模型的拓展，能够更好地建模词序信息，致力于解决语法问题，故较为适合 GEC 任务。

本文使用结构化的 skipngram 算法学习词的 wang2vec 向量表示，相较于 skip-gram 用相同的输出层预测上下文的各个词，结构化的 skipngram 针对不同相对位置的词都有一个专用的预测器，这种针对性的拓展使得 wang2vec 能够生成面向语法问题的词向量。

2.4.2.2 wang2vec 官方训练工具包

本文使用官方的 wang2vec 工具包（https://github.com/wlin12/wang2vec），他们对 word2vec 进行了简单拓展，表 2-3 列出了 wang2vec 新增参数的名称、意义、默认值与其他信息。

表 2-3 wang2vec 新增参数

参数名	意义	默认值	其他信息
-nce	噪声对比估计（NCE）也是一种 softmax 函数的近似计算方法。指定 NCE 负样本的个数	0（即不启用 NCE 机制）	启用 NCE 时，负样本个数通常的取值范围为：3-10

续表 2-3

-type	词向量学习算法	无	0：CBOW； 1：skip-gram； 2：CWINDOW； 3：结构化 skip-gram
-cap	限制参数值位于[-50, 50]区间	0（关闭该模式）	不检查数值溢出可以使得计算效率最大化，如果出现段错误，可考虑设置该值为 1，即-cap 1，代价是训练速度略微下降

2.4.3 cw2vec

2.4.3.1 cw2vec 词向量训练算法

与西方语言不同，汉字是强表意的文字，偏旁部首、字符组件和笔画特征等都带有较多语义信息，已有的许多词向量模型都是基于西方语言，不能够有效利用汉字字符内部的语义信息来增强词向量。

西方语言中，有一些工作使用 subword（子词，亚词）信息来增强词向量，例如 fasttext[39]和 CHARAGRAM[61]。在中文词向量领域，2015 年 Chen 等[62]提出了 CWE 模型，使用原词语的向量表示和该词每个字的向量做平均，得到新的词语向量，即嵌入矩阵存储了词和字的嵌入表示。Sun 等[63]在 2014 年和 Li 等[64]在 2015 年则使用偏旁部首来增强中文字向量。2017 年，Yu 等[65]提出 JWE 模型，将汉字进行更细粒度地拆分，根据人工总结的“字件”，将汉字拆分成更小的模块，融合词、汉字和字件表示来得到新的词向量。

Cw2vec 则提出了“n 元笔画”的概念，其嵌入矩阵中存储了词和 n 元笔画的向量表示，它按如下步骤进行词表示学习：

（1）将上下文的中间词拆分为笔画序列。

（2）通过不同宽度的滑动窗口得到该词的若干笔画 n-gram。

（3）将中间词的若干笔画 n-gram 向量求和作为中间词的表示，上下文中的其他词仍然使用它们的词向量表示。

（4）使用 skip-gram 模型和负采样算法学习词和 n-gram 笔画的向量表示。

（5）训练完毕后，输出嵌入矩阵存储的词向量表示即可。

cw2vec 的核心思想和 fasttext 非常类似，cw2vec 将词转化为笔画序列，而笔画 n-gram 类似于英文词的字符序列的字符 n-gram。

2.4.3.2 cw2vec 开源实现工具包

本文使用 cw2vec 开源包（https://github.com/bamtercelboo/cw2vec）完成相应 token 向量的训练，训练脚本相关的参数说明如表 2-4 所示，包括名称、意义、默认值与其他信息。

表 2-4 cw2vec 开源包参数说明

参数名	意义	默认值	其他信息
-input	同 word2vec 的-train	无	无
-infeature	笔画特征文件，即汉字和它对应的笔画序列	无	无
-output	同 word2vec 的-output	无	无
-minCount	同 word2vec 的-min-count	10	无
-minn	笔画 n-gram 的最小长度	3	无
-maxn	笔画 n-gram 的最大长度	6	训练词向量可设置为 12；BPE 向量可设为 9；字向量可设为 6
-t	同 word2vec 的-sample	0.001	有用的取值范围为：1e-3~1e-5
-lr	默认的学习率	0.05	由于 cw2vec 也是基于 skip-gram 模型，故可设置为 0.025
-lrUpdateRate	参数更新指定步后更新学习率	100	无
-dim	同 word2vec 的-size	100	按照语料规模与下游模型架构来设置
-ws	同 word2vec 的-window	5	word2vec 官方推荐 CBOW 设置为 5，skip-gram 设置为 10
-epoch	同 word2vec 的-iter	5	无
-neg	同 word2vec 的-negative	5	通常的取值范围为 3-10
-loss	负采样时使用的损失函数	ns	无
-thread	同 word2vec 的-threads	1	根据机器空闲资源酌情设置即可
-pretrainedVectors	预训练的词向量	无	无
-saveOutput	是否保存输出层参数	false	一般可以不保存

2.5 本章小结

本章主要概述了与本文相关的一些背景知识。首先介绍了 ConvS2S 模型，包括它

的编解码结构与注意力机制；接着介绍了三种 seq2seq 模型的解码生成方法，包括贪心解码、beam search 和增量式解码；然后阐述了 BPE 建模技术在 GEC 任务中的应用与意义；最后还介绍了三种词向量训练算法，包括作为基线模型的 word2vec、适用于语法问题的 wang2vec 和笔画 n-gram 信息增强的 cw2vec。

第 3 章 实验设置与数据预处理

3.1 实验软硬件环境

3.1.1 硬件设备

本实验基于一台桌面工作站和一台双卡异构 GPU 服务器完成，桌面工作站与 GPU 服务器的具体配置信息如表 3-1 和表 3-2 所示。

表 3-1 桌面工作站硬件配置

属性名	属性值
GPU	NVIDIA GeForce GTX 1080Ti
显存	11GB
CPU	Intel(R) Xeon(R) W-2135 CPU @3.70GHz（6 核 12 线程）
内存	16GB
磁盘容量	2TB

表 3-2 双卡异构 GPU 服务器硬件配置

属性名	属性值
GPU1	NVIDIA TITAN V（12GB 显存）
GPU2	NVIDIA GeForce RTX 2080Ti（11GB 显存）
CPU	Intel(R) Core(TM) i9-7920X CPU @2.90GHz（12 核 24 线程）
内存	64GB
磁盘容量	4 块磁盘（1TB+1TB+1TB+4TB）

其中，TITAN V 和 RTX 2080Ti 支持混合精度训练[66]，而 GTX 1080Ti 不支持该机制。混合精度训练表示深度神经网络的训练中混用了单精度（FP32）和半精度（FP16）浮点数类型，权重、激活值和梯度都以半精度浮点数存储，使得网络占用的内存空间缩小一半左右。由于半精度浮点数精度有限，为保留较小数量级的梯度，需在误差反向传播时，将损失函数乘上一个大于 1.0 的因子，损失缩放因子的取值一般为 8~128。

3.1.2 软件环境

本实验基于 fairseq NLP 库（https://github.com/pytorch/fairseq），fairseq 是 Facebook AI 研究院（FAIR）开源的基于 PyTorch 深度学习框架的序列建模工具包[48]，它支持 LM

和 NMT 两种文本生成 NLP 任务，支持混合精度训练，支持分布式多 GPU 卡数据并行训练，支持在 CPU 和 GPU 上进行快速 beam search 解码生成，支持做大批量训练，且灵活易于拓展。目前 fairseq 实现了三种 seq2seq NMT 基础架构：（1）LSTM seq2seq 模型（基于 Luong 在 2015 年的工作[18]）；（2）ConvS2S 模型[19]；（3）Transformer 模型[20]。

桌面工作站和双卡异构 GPU 服务器的系统环境如表 3-3 和表 3-4 所示。

表 3-3 桌面工作站系统环境

属性名	属性值
操作系统	Ubuntu 16.04.5 LTS
Python 发行版	Anaconda3-4.4.0-Linux-x86_64（Python 3.6.1）
深度学习框架	PyTorch 0.4.1
fairseq NLP 库	fairseq 0.6.0
CUDA	CUDA 9.0（cuDNN v7.3.1）

表 3-4 双卡 GPU 服务器系统环境

属性名	属性值
操作系统	Ubuntu 16.04.5 LTS
Python 发行版	Anaconda3-4.4.0-Linux-x86_64（Python 3.6.1）
深度学习框架	PyTorch 1.0.1.post2
fairseq NLP 库	fairseq 0.6.0
CUDA	CUDA 10.1（cuDNN v7.5.0）

3.2 训练数据及其预处理

本小节主要介绍实验用到的相关数据集及其预处理。NLPCC 2018 GEC 训练集（3.2.1 小节）和 HSK 平行语料（3.2.3 小节）用于训练校对模型，中文维基语料（3.2.2 小节）用于训练词向量和统计语言模型。

3.2.1 NLPCC 2018 GEC 训练集和数据处理

3.2.1.1 NLPCC 2018 GEC 训练数据介绍

NLPCC 2018 GEC 训练集（http://tcci.ccf.org.cn/conference/2018/taskdata.php）来自于 NLPCC 2018 GEC 共享任务测评，官方数据的原始格式为：

（1）每行是对一个原始输入的校对，每行可能包含 0 个（即原始输入为正确句子）、1 个或多个平行句子对，如表 3-5 所示；

（2）各行包括 4 个字段，各字段间用制表符（tab）分隔，各字段的含义如表 3-6 所示。

表 3-5 训练数据的一些示例

原始句子	目标句子
长成大人，我盒饭做的很开心。	长达成人后，我做盒饭做得很开心。
城市里的人能度过多方面的生活。	城市里的人能过丰富多彩的生活。
	城市里的人能过多种多样的生活。
	城市里的人能过多方面的生活。

表 3-6 NLPCC 2018 GEC 训练数据字段名及其含义

字段序号	字段名	字段意义
1	sens_id	句子在短文中的索引，从 1 开始
2	num_correct	目标句子的个数
3	orig_sen	原始句子
4	corrections	若干目标句子（如果 num_correct≠0）

目前中英文公开的最大规模平行校对语料（中文的 NLPCC 2018 GEC 数据集[47]，英文的 Lang-8 语料[41]）都源自于 Lang-8 网站（https://lang-8.com/），Lang-8 是一个多语言学习平台，由原生语言人士（native speakers）挑选学习者写的短文（作文）进行修改。当前公开的较大规模平行语料都以类似众包的形式得到，数据质量并不高，语料存在诸多不合规的地方，于是引出后文的训练数据预处理（3.2.1.2 小节）以及数据清洗实验（4.2.2 小节）。当前的校对平行语料主要由非原生语言学习者产生，故当前的 GEC 系统主要针对语言学习者。

3.2.1.2 NLPCC 2018 GEC 训练数据的预处理

NLPCC 2018 GEC 训练数据的预处理包括如下工作：

（1）从 NLPCC 原始训练数据移除重复行。

（2）从 NLPCC 原始训练数据移除空格。

（3）将训练数据从原始格式转为平行句子对。

（4）如果 num_correct 字段为 0，则生成目标端等同于源端的句子对，实验表明增

加恒等映射有助于提升性能。

（5）根据目标句子数生成若干平行对，以解决 num_correct 和纠正句子的实际数目不匹配的问题。

（6）若目标句子包含“不需要修改”字样，则生成目标端等同于源端的句子对。

（7）保留汉字、英文字母、数字、中英文标点符号，移除其他非法符号。

（8）使用 OpenCC（https://github.com/BYVoid/OpenCC）将繁体中文转为简体中文。

（9）控制切分粒度，使用 jieba 分词工具（https://github.com/fxsjy/jieba）切词或将句子切分为字序列。

（10）从生成的平行句子对中移除重复行。

原始训练数据有 717241 行，去除重复后有 710000 行，处理完的平行句子对有 1208721 个训练样本。

3.2.2 维基百科中文语料和数据处理

3.2.2.1 中文维基语料介绍

本文从维基 dump 网站（https://dumps.wikimedia.org）下载 XML 格式的中文维基百科语料（https://dumps.wikimedia.org/zhwiki/latest/zhwiki-latest-pages-articles.xml.bz2），XML 文件中的 text 标签内部即为文章内容，由于维基百科语料属于网络文本，需要对其进行必要的处理与清洗。

3.2.2.2 预处理中文维基语料

中文维基百科语料的处理工作包括：

（1）使用 gensim 工具包[67]（https://github.com/RaRe-Technologies/gensim）的 gensim.scripts.segment_wiki 模块将中文维基百科语料从 XML 格式转为 json 格式（wiki.json.gz）。

（2）wiki.json.gz 每行都是一个 json 字符串，具体格式如图 3-1 所示，使用 Python 的 json 库解析 json 字符串提取词条的标题与内容。

```
{
    “title”: “string”,
    “section_titles”: [“title_1”, …, “title_n”],
    “section_texts”: [“text_1”, …, “text_n”]
}
```

图 3-1 json 字符串格式

（3）保留汉字、英文字母、数字、中文标点符号，移除其他非法符号。

（4）使用 OpenCC 将繁体中文转为简体中文。

（5）控制切分粒度，使用 jieba 分词工具切词或将句子切分为字序列。

3.2.2.3 对中文维基语料应用字切分

按 3.2.2.2 小节所述处理中文维基语料，指定切分等级为字即可，处理完的维基语料用于训练中文字向量和字级别的语言模型。

预处理完毕的中文维基语料相关统计信息如表 3-7 所示。

表 3-7 字级别中文维基语料的相关统计信息

属性名	属性值	备注
行数	39108468	使用 wc -l 命令
字数	770088004	使用 wc -w 命令
文件大小	2.5GB	使用 ls -all -h 命令

3.2.2.4 对中文维基语料应用 BPE 切分

具体步骤为：

（1）基于分好词的平行语料目标端学到 BPE 模型（即 BPE 词汇表）。

（2）按 3.2.2.2 小节所述处理中文维基语料，指定切分等级为词。

（3）依据 BPE 模型对词级别的中文维基语料应用子词切分，将词序列转为 BPE 序列。

处理完的维基语料用于预训练中文 BPE token 的嵌入表示。预处理完毕的 BPE 级别中文维基语料相关统计信息如表 3-8 所示。

表 3-8 BPE 级别中文维基语料的相关统计信息

属性名	属性值
行数	39108468
BPE token 数目	561602590
文件大小	2.6GB

3.2.3 汉语水平考试平行语料

HSK（汉语水平考试的拼音缩写）语料来自北京语言大学（BLCU）的“HSK 动态

作文语料库”[68]，该语料库是 BLCU 崔希亮教授主持的一个国家汉办科研项目，项目编号：HBK01-05/023。“HSK 动态作文语料库”是母语非汉语的外国人参加高等汉语水平考试作文考试的答卷语料库，收集了 1992-2005 年的部分外国考生的作文答卷。语料库 1.0 版收入语料 10740 篇，约 400 万字，于 2006 年 12 月上线。2008 年 7 月，经修改补充，语料库 1.1 版语料总数达到 11569 篇，共计 424 万字。

本文使用的 HSK 平行数据来自 NLPCC 2018 GEC 共享任务 BLCU 团队 github 上开源的项目（https://github.com/blcu-nlp/NLPCC_2018_TASK2_GEC），该平行语料质量较高，且已经预处理完毕，共计有 156870 个平行句子对。

3.3 基准测试集和评价指标

3.3.1 基准测试集介绍

NLPCC 2018 GEC 共享评测任务的基准测试集包括源端输入与 m2 格式的官方参考编辑，源端输入共计 2000 个句子，由学习汉语的外国学生编写，测试语料由专业人士校对，官方参考编辑的格式与 CoNLL-2013 和 CoNLL-2014 基准测试集（https://www.comp.nus.edu.sg/~nlp/conll14st.html）一致。

3.3.2 校对任务评价指标

NLPCC 2018 GEC 共享任务的官方评估指标为 M^2 $F_{0.5}$ 分数[69]，与 CoNLL-13 和 CoNLL-14 英文 GEC 共享任务一致。M^2 $F_{0.5}$ 分数由式（3-1）、式（3-2）和式（3-3）计算得到：

$$P = \frac{\sum_{i=1}^{n} | e_i \cap g_i |}{\sum_{i=1}^{n} | e_i |} \tag{3-1}$$

$$R = \frac{\sum_{i=1}^{n} | e_i \cap g_i |}{\sum_{i=1}^{n} | g_i |} \tag{3-2}$$

$$F_{0.5} = \frac{(1+0.5^2) \times R \times P}{R + 0.5^2 \times P} \tag{3-3}$$

其中，P 表示校对系统的查准率；R 表示校对系统的查全率；$F_{0.5}$ 表示 GEC 任务中精确率（P）两倍重要于召回率（R）；$\{e_1,\ldots e_n\}$ 表示系统对各个句子的编辑；e_i 表示校对系统的对某个句子的编辑集合；$\{g_1,\ldots,g_n\}$ 表示各个句子的黄金编辑；g_i 表示某个句子的黄金编辑集合。

NLPCC 2018 GEC 官方性能评估指标计算流程为：

（1）移除校对系统输出 token 序列中的空格。

（2）用 pkunlp 官方分词工具（http://59.108.48.12/lcwm/pkunlp/downloads/libgrass-ui.tar.gz）重切分校对系统输出（m2 格式的官方参考编辑也是基于 pkunlp 切分句子）。

（3）将系统输出编辑与官方参考编辑进行最大匹配（MaxMatch，M^2），使用 m2scorer 性能评估脚本（https://www.comp.nus.edu.sg/~nlp/conll14st.html）计算 M^2 $F_{0.5}$ 分数。

3.4 本章小结

本章主要概述了实验的一些准备工作。首先介绍了实验的软硬件环境，包括使用的异构显卡设备与系统环境；然后介绍了实验用到的三种训练数据，包括 NLPCC 2018 GEC 训练集、维基百科中文语料和 HSK 平行语料，并分别给出它们的预处理方案；最后阐述了本实验所用的基准测试集和性能评估指标 M^2 $F_{0.5}$ 分数，并给出性能指标的计算流程。

第 4 章 基于卷积编解码网络的中文校对模型

本章首先确定基线模型（4.1 小节），然后基于数据扩增与数据清洗（4.2 小节）提升性能，接着提出一种基于字级别卷积编解码网络的中文校对模型（4.3 小节），最后基于预训练的嵌入表示提升模型性能（4.4 小节）。本章主要进行单模型相关的实验，为第 5 章和第 6 章做好准备。

4.1 基于卷积编解码网络构建中文校对基线模型

本文选择基于 ConvS2S 构建中文 GEC 模型是因为：

（1）CNN 基于大小有限的卷积核抽取特征，其捕捉局部依赖的能力强于 RNN 系列的网络（LSTM 和 GRU[15]等），且基于局部上下文信息即可纠正多数错误。此外，Chollampatt 和 Ng[1]发现结合注意力的 LSTM seq2seq 模型解码时间步会把注意力几乎全部分配给源端的某个词，这使得 LSTM 模型倾向于将词直接从源端拷贝到目标端而不是做出校对，这导致基于 LSTM 的 GEC 模型拥有更高的精度和更低的召回率。

（2）RNN 系列的网络随着时间步的增长有遗忘信息的趋势[70]，远距离 token 对当前位置的影响较小。ConvS2S 通过卷积层的堆叠，感知野亦能有效增大。并且 CNN 对句子各个部分同时进行卷积操作，由于权值共享，ConvS2S 对输入长度不敏感。

4.1.1 小节阐述基线模型的架构设置与训练细节，4.1.2 小节报告基线模型的性能表现。

4.1.1 基线模型的架构设置与训练细节

本文基于 ConvS2S 构建中文校对模型，本小节阐述基于 ConvS2S 的基线模型的具体设置，基线模型的部分架构设置与训练细节参考了 Chollampatt 和 Ng[1]的工作，如下关于基线模型的设置在本文后续实验中保持不变：（1）编码端和解码端的 token 嵌入向量维数均设置为 500，解码器的输出维数设置为 500；（2）编码器堆叠 7 层编码层，解码器堆叠 7 层解码层，编码器和解码器的隐藏状态维数均设为 1024，卷积窗口的宽度设为 3；（3）使用 NAG 优化器[71]，动量值设为 0.99，初始学习率设为 0.25，学习率衰减因子设为 0.1，当学习率衰减至小于 1e-4 时触发 early stopping 机制；（4）dropout 概率[72]设为 0.2，ConvS2S 模型将 dropout 机制应用到输入嵌入向量、各编码层的输入、各解码层的输入和解码器的输出；（5）梯度裁剪阈值设为 0.1；（6）设置一个批的数据至多包含 60 个句子；（7）从训练数据中随机切分 5000 个样本作为验证集，验证集的平行数据格式用于触发 early stopping，由平行验证集生成的 m2 格式参考编辑用于计算模型在验证集上的 M^2 $F_{0.5}$ 分数；（8）基于 NLPCC 2018 GEC 基准测试集和 M^2 $F_{0.5}$ 分数报告模型性能；（9）beam search 解码的集束宽度设置为 12；（10）基于验证集损

失从各迭代检出点中挑选最佳模型。

在本文后续实验中会发生变化的基线模型的设置如下所示：（1）仅基于 NLPCC 2018 训练集训练校对模型；（2）训练显卡为 TITAN V，启用混合精度训练（使得训练速度更快），设置初始损失缩放因子为 128；（3）字级别建模（中文 GEC 场景中字级别的句子切分粒度应当更为自然），源端和目标端词汇表规模均设置为 6000；（4）随机初始化字向量嵌入矩阵；（5）随机数种子设置为 1。

4.1.2 基线模型的性能表现

基线模型的设置如 4.1.1 小节所述，表 4-1 报告了基线模型在基准测试集上的性能表现。

表 4-1 基线模型的性能表现（基于基准测试集）

模型	精确率	召回率	M^2 $F_{0.5}$
基于 ConvS2S 的 CGEC 基线模型	36.89	9.19	23.02

由表 4-1 的结果可知，基线模型的性能表现并不理想，故本文后续将通过一系列的方法提升模型性能。

4.2 基于数据扩增与数据清洗提升中文校对模型的性能

本文基于如下的一些观点进行数据扩增和数据清洗：

（1）更多的数据应当能够有更好的性能。本文将 NLPCC 2018 GEC 训练数据与 HSK 平行语料（见 3.2.3 小节）进行融合，HSK 平行语料来自北语的“HSK 动态作文语料库”。

（2）质量更高的数据应当能够有更好的性能。当前公开的较大规模校对平行语料都以类似众包的形式得到（例如从 Lang-8 采集），通过数据清洗实验过滤平行语料中的噪声样本。

4.2.1 小节进行数据扩增验证性实验（基于测试集报告模型性能），4.2.2 小节为数据清洗对比实验（基于验证集报告性能）。

4.2.1 数据扩增

数据扩增实验除了训练数据与训练显卡有所不同，其他配置与 4.1.1 小节保持一致。数据扩增实验结果如表 4-2 所示。

对表 4-2 的一些说明：

（1）“融合模式”为 NLPCC 表示仅使用 NLPCC 2018 GEC 训练集（去除重复后

表 4-2 数据扩增实验结果（基于基准测试集）

融合模式	训练显卡	混合精度训练设置	精确率	召回率	M^2 $F_{0.5}$
NLPCC	TITAN	fp16-loss scale=128	36.89	9.19	23.02
NLPCC+HSK	1080Ti	nofp16	44.90	11.83	**28.80**
NLPCC+HSK	2080Ti	fp16-loss scale=8	44.64	11.40	**28.19**
NLPCC+HSK	TITAN	fp16-loss scale=8	43.12	11.67	**28.02**

共计 1208721 个平行句子对），为 NLPCC+HSK 表示将 NLPCC 2018 GEC 训练集与 HSK 语料堆叠融合（去除重复后共计 1354287 个训练样本）。

（2）“训练显卡”为 TITAN 表示基于 NVIDIA TITAN V 训练 ConvS2S CGEC 模型，为 1080Ti 表示基于 GTX 1080Ti 训练，为 2080Ti 表示使用 RTX 2080Ti。

（3）“混合精度训练设置”中，fp16 表示启用混合精度训练机制，nofp16 表示不启用（1080Ti 显卡不支持该机制），loss scale 为混合精度训练时初始的损失缩放因子。

表 4-2 的实验结果表明：

（1）融合 HSK 语料能显著增强模型性能（$F_{0.5}$ 值从 23 提升至 28），这与我们的直觉相符合，故本文后续实验都基于 NLPCC+HSK 融合数据。

（2）不同架构的显卡和混合精度训练对模型的性能有一定影响。

本文在实验过程中发现，相同模型架构、相同超参设置（随机数种子亦相同）、相同数据供给的实验在 TITAN V、2080Ti 和 1080Ti 三张显卡上获得了不同的性能表现，于是本文又通过一些异构显卡对比实验（未完全在本文中给出）发现：

（1）相同配置的实验在不同显卡上的 $F_{0.5}$ 值有 1 以内的波动，且未能发现存在明显规律，故本文后续实验还将报告训练显卡，且对比实验需基于同一显卡，以确保实验只有单一变量。

（2）loss scale 对性能亦有一定影响，2080Ti-fp16-loss scale=8 比 2080Ti-fp16-loss scale=128 性能高约 0.1，TITAN-fp16-loss scale=8 与 TITAN-nofp16 结果几乎一致，故本文后续实验混合精度训练的初始损失缩放因子均设置为 8。

（3）相较于 loss scale=128，loss scale=8 会减缓收敛速度。

4.2.2 数据清洗

数据清洗实验通过不同的阈值从融合数据中移除源端或目标端过长的句子对（remove-long 实验，4.2.2.1 小节）、目标端长度远大于源端的句子对（remove-high 实验，4.2.2.2 小节）和目标端长度远小于源端的句子对（remove-low 实验，4.2.2.3 小节）。数据清洗实验中句子长度为句子的字数。

4.2.2.1 remove-long 实验

remove-long 实验将融合数据中源端或目标端长度大于阈值的句子对移除，通过对比实验确定最佳阈值，表 4-3 和图 4-1 报告了模型在验证集上的性能表现。

表 4-3 不同阈值的 remove-long 实验结果（基于验证集，基于 1080Ti-nofp16）

过滤长句的阈值	精确率	召回率	M^2 $F_{0.5}$
long=1000	37.63	11.22	25.58
long=140	38.35	9.72	24.13
long=120	37.61	11.50	25.86
long=100	36.11	11.18	24.98
long=80	37.00	11.06	25.18

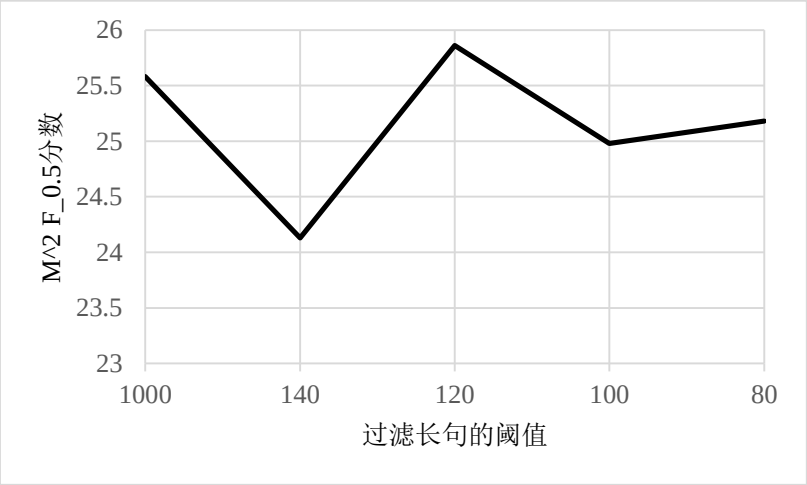

图 4-1 不同过滤长句阈值对 $F_{0.5}$ 值的影响（基于验证集）

由表 4-3 和图 4-1 的实验结果可以发现：随着阈值的减少（long=1000 表示不移除任何长句），基于验证集的 $F_{0.5}$ 值未表现出先增大后减小的规律，故本文后续实验都不移除任何长句。Chollampatt 和 Ng[1]在数据预处理时过滤长度大于 80 的英文句子，Ji 等[23]则移除长度大于 100 的英文句子，但本文发现移除长句并不能带来性能提升，因为 ConvS2S 模型对输入句子的长度并不敏感。

4.2.2.2 remove-high 实验

remove-high 实验将融合数据中目标端与源端长度比大于阈值的句子对移除，通过对比实验确定最佳阈值，表 4-4 和图 4-2 报告了模型在验证集上的性能表现。

表 4-4 不同阈值的 remove-high 实验结果（基于验证集，基于 2080Ti-fp16-loss scale=8）

过滤阈值	精确率	召回率	M^2 $F_{0.5}$
high=200.0	37.20	11.05	25.25
high=9.0	35.31	10.92	24.40
high=7.0	38.35	11.89	26.54
high=5.0	36.96	11.18	25.30
high=3.0	37.81	11.56	26.00

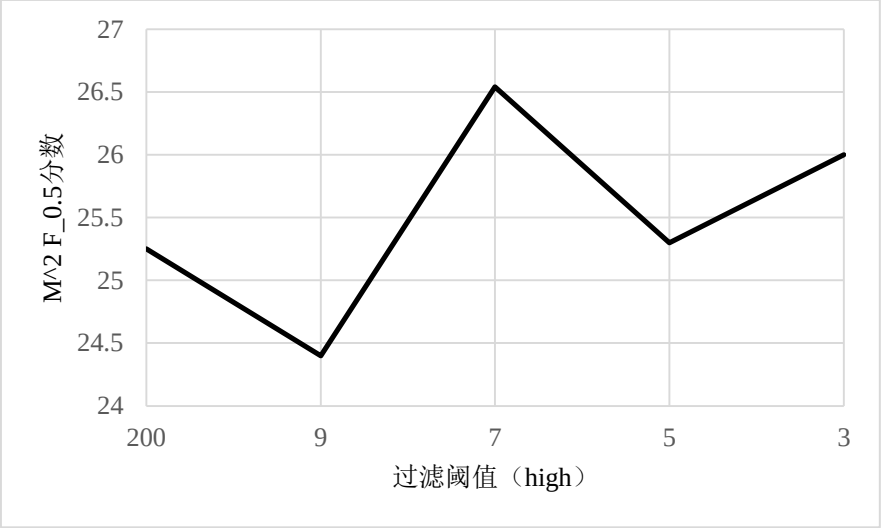

图 4-2 high 的不同取值对 $F_{0.5}$ 值的影响（基于验证集）

从表 4-4 和图 4-2 的实验结果可以发现：随着阈值的减少（high=200.0 表示不移除任何句子对），基于验证集的 $F_{0.5}$ 值未表现出先增大后减小的规律，故本文后续实验不移除目标端长度远大于源端的句子对。Chollampatt 和 Ng[1]在数据预处理时设置 high=9，但本文发现 remove-high 并不能带来性能提升。

4.2.2.3 remove-low 实验

remove-low 实验将融合数据中目标端与源端长度比小于阈值的句子对移除，通过对比实验确定最佳阈值，表 4-5 和图 4-3 报告了模型在验证集上的性能表现。

从表 4-5 和图 4-3 的实验结果可以发现：当 low=0.1 时，$F_{0.5}$ 值达到峰值，存在波峰结构（即随着阈值的增大，$F_{0.5}$ 值先增大后减小），故本文后续实验都移除目标端与源端长度比低于 0.1 的句子对。由于平行语料存在许多噪声样本，过滤目标端长度和源端差距悬殊的句子对能有效提升模型性能，基于 TITAN V 训练的 ConvS2S CGEC 模型在基准测试集上的 $F_{0.5}$ 值从 28.02（不移除句子）上升至 28.53（low=0.1）。

表 4-5 不同阈值的 remove-low 实验结果（基于验证集，基于 TITAN-fp16-loss scale=8）

移除阈值	精确率	召回率	M^2 $F_{0.5}$
low=0.0	37.00	11.33	25.46
low=0.1	**37.53**	**11.36**	**25.69**
low=0.2	37.50	11.34	25.65
low=0.3	36.27	10.97	24.82
low=0.4	36.25	11.25	25.10
low=0.5	37.26	10.95	25.16

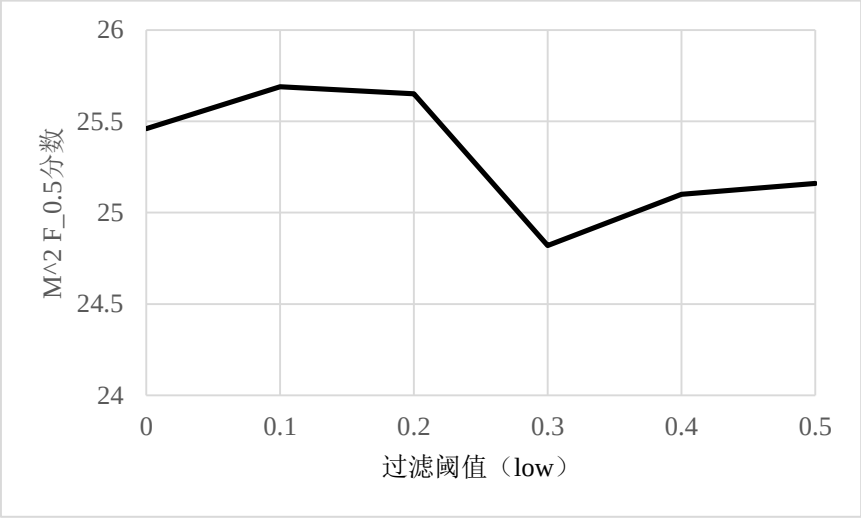

图 4-3 low 的不同取值对 $F_{0.5}$ 值的影响（基于验证集）

4.3 基于字级别卷积编解码网络的中文校对模型

本文基于如下的一些观点构建字级别的校对模型：

（1）现有的分词算法大都针对正常文本，将标准的分词算法应用到 CGEC 场景，对有错误的文本进行分词会导致误切分。此外，分词算法本身也存在歧义切分和未登录词的问题，即分词过程可能还会引入额外的错误信息。

（2）现有的词级别 NMT 模型大都通过限制词汇表规模来缓解 softmax 函数计算量大的问题，有限容量词汇表会导致罕见词变为 OOV token，而输入输出句子对中带有<UNK>符号势必会导致性能损失。

（3）尽管子词切分技术能有效解决 NMT 模型受限于罕见词的问题，但 BPE 序列的构建依然需要对句子进行分词。

（4）汉字字符本身就带有语义信息，故中文 NLP 任务中字级别的建模也很自然。

4.3.1 基于不同级别建模的中文文本自动校对

中文 NLP 任务的建模级别有字、词和子词，故本小节将进行不同级别建模的对比实验，实验结果见 4.3.2 小节。

4.3.1.1 字级别的卷积编解码网络中文校对模型

字级别的 ConvS2S CGEC 模型与 4.2.2.3 小节 low=0.1 的模型一致，故将直接复用之前的实验结果。

4.3.1.2 词级别的卷积编解码网络中文校对模型

词级别建模的特殊设置如下：

（1）基于结巴分词工具切分训练句子对；（2）过滤出现次数<=3 的低频词，处理后的源端词汇表规模为 69677，目标端词汇表规模为 65591；（3）词级别的模型由于嵌入矩阵和 softmax 层参数过多，显存占用较大，故设置一个批的数据至多包含 30 个句子。

4.3.1.3 子词级别的卷积编解码网络中文校对模型

BPE 级别建模的特殊设置如下：（1）基于结巴分词切分训练句子对，应用 BPE 编码构造 BPE 序列；（2）设置源端和目标端词汇表规模均为 37000。

4.3.2 不同级别建模对比实验结果

不同级别建模的对比实验结果如表 4-6 所示，同时报告模型在验证集和测试集上的性能表现。

表 4-6 不同级别建模对比实验结果（基于 TITAN-fp16-loss scale=8）

模型架构	精确率（dev）	召回率（dev）	M^2 $F_{0.5}$（dev）	M^2 $F_{0.5}$（test）
ConvS2S+Char+Random	37.53	11.36	**25.69**	**28.53**
ConvS2S+Word+Random	25.45	10.28	19.65	20.01
ConvS2S+BPE+Random	31.17	11.66	23.35	27.80

表 4-6 中的 Random 表示随机初始化嵌入矩阵，根据表 4-6 的实验结果可以发现：

（1）词级别的校对模型性能远弱于字和子词级别，主要原因在于词汇表规模有限且现有的分词算法与 CGEC 场景不匹配。

（2）BPE 级别的模型远优于词级别的模型，主要原因是 BPE 技术能有效解决 NMT

模型受限于罕见词的问题。

（3）字级别的模型优于 BPE 级别的模型，主要是因为 BPE 建模依赖句子的显示分词，这说明字级别的模型更为适合 CGEC 任务。

（4）BPE 级别模型的性能与字级别模型较为接近，故 4.4 小节还做了预训练的词向量增强 BPE 级别校对模型的对比实验。

4.4 基于预训练的词向量提升中文校对模型的性能

在自然语言处理领域中，预训练的词向量和语言模型都属于迁移学习，即将从 A 任务学到的知识迁移到 B 任务中。根据文献[28]，相较于随机初始化嵌入矩阵，使用预训练的词向量有助于提升词和 BPE 级别校对模型的性能，然而 Ren 等[28]的工作直接使用 wang2vec 而未系统地给出不同词向量的对比结果。此外，关于预训练中文字向量增强 CGEC 模型的研究也知之甚少，于是本文基于词向量的基线模型 word2vec、专门针对语法问题的 wang2vec 和笔画信息增强的 cw2vec，训练了字和 BPE 级别 token 的嵌入表示（4.4.1 小节），通过对比实验（4.4.2 小节）试图找到有益于字和 BPE 级别 CGEC 模型的预训练嵌入表示。

4.4.1 基于不同词向量算法训练不同级别的嵌入表示

本小节将基于三种词向量算法训练字和 BPE 级别 token 的嵌入表示，将维基百科中文语料按 3.2.2.3 小节所述进行字切分用于训练字向量，将中文维基语料按 3.2.2.4 小节所述进行 BPE 切分用于预训练 BPE 向量。

4.4.1.1 基于 word2vec 训练字和 BPE 的嵌入表示

训练字和 BPE 表示的命令除了训练语料与结果向量存储文件不同，其余完全相同，依据 2.4.1.3 小节对 word2vec 工具的介绍可知训练命令为：

./word2vec -train wiki.zh.seg.txt -output vec.500d.txt -size 500 -window 10 -sample 1e-4 -hs 0 -negative 10 -threads 10 -iter 5 -binary 0 -cbow 0

训练命令的含义为：

（1）预训练的字和 BPE 向量维数均为 500，这是因为 ConvS2S 模型编码端和解码端 token 的嵌入向量大小为 500。

（2）使用 skip-gram 模型，最大预测跨度设为 10。

（3）使用负采样，负样本数设为 10，触发下采样的词频阈值设为 1e-4。

（4）使用 10 个线程并行加速训练过程，至多进行 5 轮迭代，输出为词向量标准文本格式。

4.4.1.2 基于 wang2vec 训练字和 BPE 的嵌入表示

训练字和 BPE 表示的命令除了训练语料与结果向量存储文件不同，其余完全相同，依据 2.4.2.2 小节对 wang2vec 工具的介绍可知训练命令为：

./word2vec -train wiki.zh.seg.txt -output vec.500d.txt -size 500 -window 10 -sample 1e-4 -hs 0 -negative 10 -nce 0 -threads 10 -iter 5 -binary 0 -type 3 -cap 0

训练命令的含义基本同 4.4.1.1 小节，此处仅介绍不同的设置：

（1）使用结构化 skip-gram 模型，这使得词向量学习算法对位置敏感，能够更好地建模词序信息，输出针对语法问题的词向量。

（2）为使得计算效率最大化，不启用数值溢出检查机制。

4.4.1.3 基于 cw2vec 训练字和 BPE 的嵌入表示

基于 cw2vec 训练字和 BPE 表示的命令除了训练语料、结果向量存储文件和笔画 n-gram 最大长度不同，其余完全相同，依据 2.4.3.2 小节对 cw2vec 工具的介绍可知训练命令为：

./word2vec substoke -input wiki.zh.seg.txt -infeature zh.chars.strokes -output vec.500d.txt -lr 0.025 -dim 500 -ws 10 -epoch 5 -minCount 10 -neg 10 -loss ns -minn 3 -maxn 6 -thread 10 -t 1e-4 -lrUpdateRate 100

训练命令的含义为：

（1）预训练的字和 BPE 向量维数均为 500。

（2）使用 cw2vec skip-gram 模型，最大预测跨度设为 10，初始学习率设为 0.025，设置每 100 步更新一次学习率。

（3）使用负采样，负样本数设为 10，触发下采样的词频阈值设为 1e-4。

（4）使用 10 个线程并行加速训练过程，至多进行 5 轮迭代，输出为词向量标准文本格式。

（5）指定笔画特征文件。

（6）设置词汇表构建时的词频过滤阈值为 10。

（7）设置笔画 n-gram 的最小长度为 3，字向量笔画 n-gram 最大长度设为 6，BPE 向量的笔画 n-gram 最大长度设为 9。

4.4.2 基于不同的嵌入矩阵初始化方式的中文文本自动校对

由于校对任务使用错误句子到正确句子的平行语料，故本文设置源端和目标端不共享词汇表，然后通过不同的方式初始化源和目标端的嵌入矩阵。

1. 随机初始化嵌入矩阵

做法如下：调用 fairseq 的训练脚本时，不使用--encoder-embed-path 和--decoder-embed-path 参数即可。

2. 基于预训练词向量的中文校对模型

具体流程为：

（1）字级别的模型选用字向量文件，BPE 级别的模型选用 BPE 向量文件，这些向量文件为词向量训练算法的输出。

（2）调用 fairseq 的训练脚本时，通过--encoder-embed-path 和--decoder-embed-path 参数指定源端和目标端使用的 token 向量文件。

（3）根据预训练的嵌入表示初始化嵌入矩阵，嵌入矩阵在后续训练中与模型其他参数一同被更新。

4.4.3 嵌入矩阵不同初始化方式对比实验结果

不同方式初始化字级别校对模型嵌入矩阵的对比实验如表 4-7 和图 4-4 所示，同时报告模型在验证集和基准测试集上的表现。

表 4-7 字级别模型不同方式初始化嵌入矩阵的对比实验（基于 TITAN-fp16-loss scale=8）

模型架构	M^2 $F_{0.5}$（dev）	M^2 $F_{0.5}$（test）
ConvS2S+Char+Random	25.69	28.53
ConvS2S+Char+Word2vec	25.71	**29.95**
ConvS2S+Char+Wang2vec	25.89	28.39
ConvS2S+Char+Cw2vec	25.55	28.22

根据表 4-7 和图 4-4 的实验结果可以发现，使用预训练的 word2vec 字向量初始化嵌入矩阵能够显著提升字级别校对模型的性能，而 wang2vec 和 cw2vec 不能给字级别的校对模型带来性能增益（甚至使得性能下降）。

注意到模型在验证集上的表现与测试集较为不匹配，验证集分数几乎没有起伏，而测试集分数起伏较大。由于官方未提供 m2 格式的验证集，故本文使用 m2scorer 项目（https://github.com/nusnlp/m2scorer）中的 scripts/edit_creator.py 脚本来将验证集从平行数据转为 m2 格式。相比较于标注人员针对源端输入手工标注的黄金编辑，通过 edit_creator.py 脚本得到的参考编辑不是最优的，所以本文的某些实验同时报告模型在验证集和测试集上的表现，当二者有冲突时以测试集为主。英文由于有 CoNLL-13 和

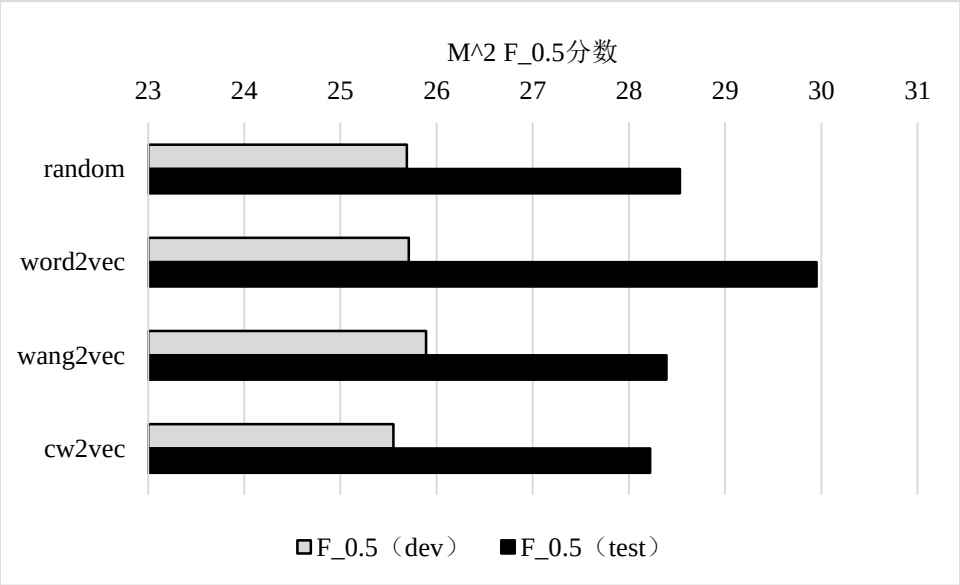

图 4-4 不同词向量对字级别模型 $F_{0.5}$ 值的影响

CoNLL-14 两次 GEC 共享任务，故英文 GEC 任务一般都将 CoNLL-13 的测试集作为验证集，将 CoNLL-14 的测试集作为基准测试集。

由于 BPE 级别模型性能与字级别较为接近，且之前的一些工作也表明预训练的 BPE 向量能提升 BPE 级别模型的性能，故本文进行 BPE 级别校对模型嵌入矩阵不同初始化方式的对比实验，探索先验知识增强的 BPE 级别模型能否超过前述 word2vec 增强的字级别校对模型，对比实验结果如表 4-8 和图 4-5 所示，同时报告模型在验证集和基准测试集上的表现。

表 4-8 BPE 级别模型嵌入矩阵不同初始化方式对比实验（基于 2080Ti-fp16-loss scale=8）

模型架构	M^2 $F_{0.5}$（dev）	M^2 $F_{0.5}$（test）
ConvS2S+BPE+Random	23.59	27.99
ConvS2S+BPE+Word2vec	**24.82**	**28.83**
ConvS2S+BPE+Wang2vec	23.15	28.67
ConvS2S+BPE+Cw2vec	24.15	28.31

根据表 4-8 和图 4-5 的实验结果可以发现：使用预训练的 BPE word2vec 和 wang2vec 嵌入表示均能有效提升 BPE 级别模型的性能，使用 cw2vec 能略微增强 BPE 级别模型的性能。尽管 word2vec 增强的 BPE 级别校对模型性能未能超过 word2vec 增强的字级别模型，本文在 6.2.2 小节基于多通道融合与重排序的 CGEC 中还是使用它来获取子词级别的信息。

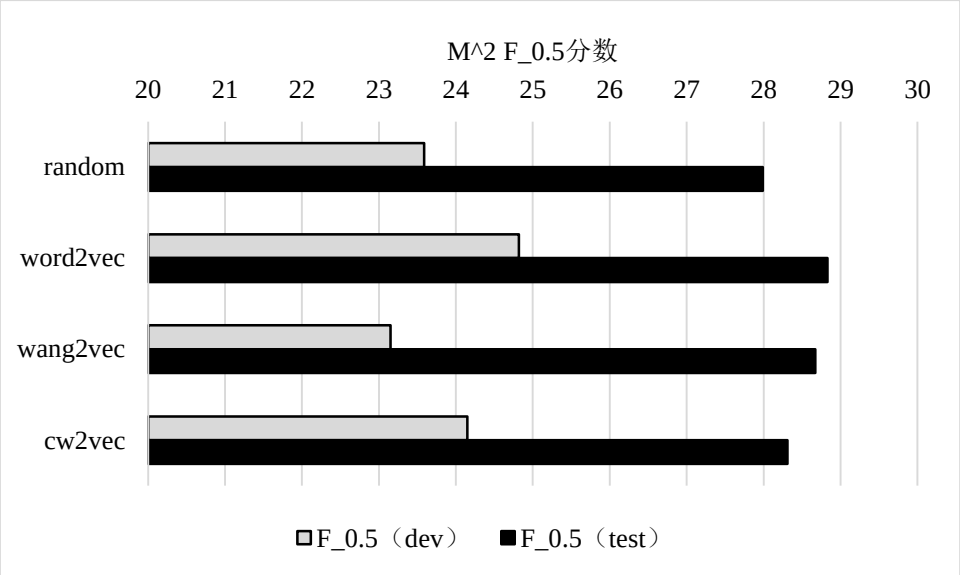

图 4-5 不同词向量对 BPE 级别模型 $F_{0.5}$ 值的影响

4.5 本章小结

本章主要进行单模型相关的实验，首先确定了基线模型，然后基于数据扩增与数据清洗提升性能，接着提出一种基于字级别卷积编解码网络的中文校对模型，最后基于预训练的嵌入表示提升模型性能。

第 5 章 基于集成解码和重排序的中文文本自动校对

当前主流的 GEC 方法基本基于 NMT 模型，而 NMT 模型基于 beam search 解码生成目标序列，这会输出解码器最终时间步得分最高的句子。由于当前公开的校对平行语料仅为中等规模，且 NMT 模型的编码器和解码器训练难度不对等[73]，这可能导致 GEC 模型的解码器训练不够充分，故仅依靠解码器分数从 n-best 列表中选择的目标输出可能并不是最优的。综上所述，对于文本校对任务而言，依据重排序组件对 NMT 模型的 n-best 输出进行重打分是非常有必要的。

重排序方法的应用会导致校对系统的精确率下降，为缓解精度损失，本文针对 4 模型集成解码的 n-best 输出应用重排序机制，这是因为集成解码能够显著提升系统的精确率。

本章首先概述了集成解码方法（5.1 小节），随后提出三种针对中文校对模型 n-best 输出应用重排序的方法（5.2 小节），接着提出基于集成解码和重排序的中文校对模型（5.3 小节），最后是相关的实验设计与结果分析（5.4 小节）。

5.1 集成解码方法概述

集成解码是一种简单有效的提升 seq2seq 模型性能的方法，它能显著提升模型精度，并进而提升性能。基于集成解码增强校对模型的流程为：

（1）单模型架构选择 ConvS2S，基于字或 BPE 级别建模，嵌入矩阵随机初始化或基于 word2vec 预训练嵌入表示，训练数据为 NLPCC+HSK 融合语料，移除目标端与源端长度比低于 0.1 的异常句子对。

（2）设置不同的随机数种子，基于不同的网络初始化训练得到多个架构相同参数不同的模型。一般的做法为 4 模型集成，各单模型的随机数种子分别设为 1、2、3、4 或 1、1000、2000、3000。

（3）beam search 增量式解码的时候：①将前一个时刻的预测输出回馈入四个单模型的解码器，从缓存恢复之前的隐藏状态；②各解码器做一步计算并用最新状态更新各自的缓存，然后各模型将当前解码时间步的隐藏状态馈入 softmax 输出层；③基于种子平均的方法，将多个模型对当前时间步的预测分布求平均，计算当前时间步各假设词的集成对数概率分数。

（4）实践使用中，只需设置 fairseq NLP 库使用指定目录下的多个模型进行集成解码即可，各模型的路径用冒号（“：”）进行分隔。

不同随机数种子初始化的单模型和集成解码的性能对比实验见 5.4.2 小节。

5.2 基于重排序的中文文本自动校对

编解码网络的编码器和解码器训练难度并不对等，这是因为编码器只需抽到源端输入句子相应的信息即可，而解码器需要一个字一个字地去生成目标序列。NMT 模型的解码器本质上是一个语言模型，而训练一个低困惑度的语言模型需要句子数为千万级别的单语语料[74]，然而目前可用的校对平行语料样本数只有百万量级，这会导致解码器训练不够充分。

NMT 模型主要基于数据增强解决解码器训练不充分的问题，例如单语增强[73]和反向翻译[79]等。基于 NMT 模型的文本校对也可以利用数据增强提升模型性能，例如 Ge 等[24]和 Ren 等[28]向训练语料扩增正确到正确的句子对（即源端和目标端相同的句子对），此外，Ge 等[24]还提出了三种基于动态数据增强的流畅度提升学习机制。由于增多训练语料会显著提升模型训练时间，而本文基于单卡训练校对模型，故本文选择使用重排序方法来缓解解码器训练不充分的问题。

本小节提出三种针对中文校对模型 n-best 输出应用重排序的方法，分别为基于编辑操作特征的 EO 重排序（5.2.1 小节）、基于语言模型特征的 LM 重排序（5.2.2 小节）和同时应用编辑操作与语言模型特征的 EO+LM 重排序（5.2.3 小节），基于前述的重排序机制增强中文校对模型，实验设计与结果分析见 5.4 小节。

5.2.1 基于编辑操作特征的重排序

5.2.1.1 编辑操作特征

AMU14[75]使用基于词的编辑距离来衡量源输入和目标输出的差异，并将源和目标端间的编辑距离作为翻译模型的特征，AMU16[6]则基于编辑距离矩阵构造编辑操作计数特征，包括将源输入转换为目标输出所需的删除、插入和替换操作的个数，这些计数的求和会等于源输入和目标输出的编辑距离，如表 5-1 所示。最小编辑操作特征能够为校对系统引入额外的信息（即校对结果与源输入的差异程度[6]），使得倾向于选择那些做出合理编辑次数的系统输出。编辑操作特征的重要程度（即特征权重）基于 MERT 算法[34]训练得到。这种针对校对任务的特征与额外信息的引入可以指导校对系统的翻译过程。

表 5-1 基于 token 的编辑距离与编辑操作计数

源端句子	目标句子	LD	D	I	S
天/ 气/ 真/ 好/ 。	昨/ 天/ 天/ 气/ 很/ 差/ 。	4	0	2	2
完/ 全/ 一/ 样/ 。	完/ 全/ 一/ 样/ 。	0	0	0	0

表 5-1 中的 LD 表示 Levenshtein distance，一般也称之为编辑距离；D 为删除操作的次数；I 为插入操作的次数；S 为替换操作的次数。

5.2.1.2 重排序机制的总体框架

重排序机制基于对数线性（log-linear）框架，结合若干特征，对 beam search 最终时间步的 n-best 列表（n 为集束宽度）重打分，给定源输入句子 S，校对候选 T 的分数由式（5-1）计算得到：

$$\text{score}(T,S)=\sum_{i=1}^{F}\lambda_i f_i(T,S) \tag{5-1}$$

其中，λ_i 表示特征权重，由 MERT 算法训练得到；f_i 表示特征函数；F 为特征数目。

对式（5-1）的一些说明：

（1）若仅使用校对模型解码器分数和编辑操作特征（三种特征，删除、插入和替换操作的次数），则为 EO 重排序。

（2）若仅使用校对模型解码器分数和语言模型特征（两种特征，语言模型分数与校对候选的词数，具体见 5.2.2 小节），则为 LM 重排序。

（3）若同时使用校对模型解码器分数、编辑操作特征和语言模型特征，则为 EO+LM 重排序。

（4）校对模型解码器分数即解码器计算的目标句子流畅度评分，为目标句子各个词对数条件概率（各解码时间步会计算条件分布）的累加。

5.2.1.3 编辑操作特征重排序的具体流程

本文基于 nbest-reranker 工具包（https://github.com/nusnlp/nbest-reranker）对校对模型的 n-best 输出列表应用重排序机制。基于编辑操作特征的重排序流程为：

（1）启用 n-best 输出机制。设置 fairseq 的 interactive.py 的--nbest 参数为集束宽度 n，使得解码生成时输出 beam search 最终时间步的 n-best 列表。

（2）输出格式转换。根据“O”和“H”标志从 interactive.py 的输出中提取源端输入的 n-best 输出及其对应的校对模型解码器分数，将输出格式转为 Moses[76]格式，Moses 格式如图 5-1 所示。

（3）获取特征。nbest-reranker 工具包的 augmenter.py 脚本相关的参数说明如表 5-2 所示，通过-i 参数接收第（2）步的输出，通过设置-f 参数指定使用编辑操作特征，并将获取到的三种编辑操作特征写入由-o 参数指定的文件，输出文件的格式如图 5-2 所示。

（4）训练重打分器。nbest-reranker 工具包的 train.py 脚本相关的参数说明如表 5-3 所示，通过-i 参数接收第（3）步的输出，-r 则用于指定参考编辑，通过-m 参数设置特

第1个 n-best输出
0 ||| 系统假设输出1 ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
0 ||| 系统假设输出2 ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
...
0 ||| 系统假设输出*n* ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
1 ||| 系统假设输出1 ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
...
1 ||| 系统假设输出*n* ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
...
第*m*+1个 n-best输出
m ||| 系统假设输出1 ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数
...
m ||| 系统假设输出*n* ||| F0= NMT模型解码器分数 ||| NMT模型解码器分数

图 5-1 Moses 格式示意图

表 5-2 augmenter.py 脚本参数说明

参数名	含义
-s 或--source-sentence-file	包含全部源端输入的文件
-i 或--input-nbest	第（2）步得到的 Moses 格式输出
-o 或--output-nbest	新增特征的 n-best 输出
-f 或--feature	各 feature 类的构造器语句

第1个 n-best输出
0 ||| 系统假设输出1 ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
0 ||| 系统假设输出2 ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
...
0 ||| 系统假设输出*n* ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
1 ||| 系统假设输出1 ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
...
1 ||| 系统假设输出*n* ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
...
第*m*+1个 n-best输出
m ||| 系统假设输出1 ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数
...
m ||| 系统假设输出*n* ||| F0= NMT解码器分数 EditOps0= #D #I #S ||| NMT解码器分数

图 5-2 新增三个编辑操作特征的 n-best 输出格式

征权重的调优度量为验证集的 M^2 $F_{0.5}$ 分数，通过--moses-dir 指定 Moses 的路径。本文使用 train.py 脚本基于 MERT 算法训练特征权重，MERT 算法能针对评估指标对特征权重进行调优，其实现来自 Moses 工具包（https://github.com/moses-smt/mosesdecoder）。

（5）应用重排序机制。基于 nbest-reranker 工具包的 rerank.py 脚本、第（3）步得到的特征分数和第（4）步得到的特征权重对校对模型的 n-best 输出进行重打分，将分

数最高的句子作为最佳校对输出。

表 5-3 train.py 脚本参数说明

参数名	含义
-i 或--input-nbest	新增特征的 nbest 输出
-r 或--reference-files	本文中为 m2 格式的参考编辑
-c 或--config	配置文件
-o 或--output-dir	输出目录
-t 或--threads	MERT 训练时使用的线程数
-a 或--tuning-algorithm	使用的权重调优算法，本文中使用 mert
-m 或--tuning-metric	调优度量，支持 bleu 和 m2
--moses-dir	Moses 项目的路径

5.2.2 基于语言模型特征的重排序

由于 GEC 本质上是一个流畅度提升的任务，故针对校对模型解码器 n-best 输出的流畅程度进行重打分能提升性能。LM 重排序基于外部语言模型增强，通过特征权重将统计语言模型计算得到的候选假设的语言模型分数、目标句子的词数和校对模型解码器分数结合，对 n-best 输出重打分，选择得分最高的句子作为校对系统输出。

本节将介绍 N 元语言模型、KenLM[77]语言建模工具包（https://github.com/kpu/kenlm）和语言模型重排序的具体流程。

5.2.2.1 统计 N 元语言模型

1. 维数灾难与 N 元语法模型

语言模型用于计算句子的出现概率，即 token 序列的联合概率，如式（5-2）所示：

$$P(S)=P(w_1,w_2,...,w_n)=p(w_1)\times p(w_2\mid w_1)\times...\times p(w_n\mid w_1,w_2,...,w_{n-1}) \quad (5\text{-}2)$$

其中，S 为句子；$\{w_1,w_2,...,w_n\}$为 token 序列。

式（5-2）为语言模型的基本形态，按乘法定理展开可以发现，模型自由参数过多，导致模型泛化能力欠缺，未在训练语料中出现过句子，按式（5-2）计算的概率一般为 0。语言模型的基本形态发生了维数灾难，维数灾难是指特征维数过多导致性能下降。

减少维数可以有效解决维数灾难问题，于是可以使用 N-gram 语言模型近似计算句子概率，N-gram 模型通过拼接重叠短语构成句子，用多维随机变量刻画短语（N 元组）而不是句子。N-gram 语言模型基于 N-1 阶马尔可夫链，它认为当前词只与前 N-1 个

token 有关，其计算词语序列构成句子的概率如式（5-3）所示：

$$P(S)=P(w_1,w_2,...,w_k)=\prod_{i=1}^{k}p(w_i\mid w_{i-n+1},...,w_{i-1}) \quad (5\text{-}3)$$

其中，$\{w_1,w_2,...,w_k\}$为 token 序列；n 即 N-gram 的长度。

2. 数据稀疏与平滑算法

相较于语言模型的基本形态，N-gram LM 的模型参数显著减少，泛化能力明显提升，但受限于训练语料的规模，数据稀疏的问题依然存在，故需要采用数据平滑的方法避免值为零的模型参数（也就是对数条件概率）出现。

数据平滑算法通过提高低（零）概率，降低高概率，使得概率分布趋于平滑。KenLM 基于修改的 Kneser-Ney（KN）平滑算法[78]解决数据稀疏问题，修改的 KN 算法是目前效果最好的平滑方法之一。

3. 语言模型分数

N-gram 语言模型分数即句子中各个 N-gram 对应的对数条件概率累加，N-gram LM 分数由式（5-4）计算得到：

$$\text{lm\_score}(S)=\lg P(w_1,w_2,...,w_k)=\sum_{i=1}^{k}\lg p(w_i\mid w_{i-n+1},...,w_{i-1}) \quad (5\text{-}4)$$

其中 lg 表示以 10 为底的对数。

5.2.2.2 KenLM 语言建模工具包

1. 估计语言模型参数

KenLM 基于修改的 KN 平滑算法估计语言模型参数，通过多线程加速 LM 的训练过程，其训练命令为：

bin/lmplz -o 5 -S 60% -T /tmp <train.txt >5gram.arpa

训练命令的含义为：

（1）设置-o 参数，训练 5-gram 语言模型。

（2）设置-S 参数，指定 KenLM 至多使用 60%的内存空间。如果分配的内存空间不够，KenLM 会自动执行基于磁盘的聚合排序。

（3）设置-T 参数，指定/tmp 为临时文件存放路径。

（4）KenLM 默认从标准输入读取训练语料，此处将输入重定向为已切分语料。按字切分则训练字级别的 LM，按词切分则训练词级别的语言模型。

（5）KenLM 默认会将结果输出到标准输出，此处将输出重定向到 arpa 后缀的文件。ARPA 是回退（backoff）N-gram LM 的存储格式，若指定 N 元组不存在，则回退

用 N-1 元语法进行参数估计。遵循 LM 的惯例，存储以 10 为底概率的对数值。

2. 剪枝

训练过程中将计数小于阈值的 N 元组从 LM 中移除。默认不进行剪枝，通过--prune 参数使能剪枝机制，它接收空格分隔的一系列非下降数字（如 0 1 1 2），每个数字对应一个阶位的 N 元组（例如第二个数字对应 2-gram），由于当前不支持一元剪枝，故第一个数字总是设置为 0，同时最后一个数字会被自动拓展到更高阶位的 N-gram。

3. 过滤

基于 bin/filter 可执行程序，针对测试集过滤语言模型，将计算测试集 LM 分数时不需要的 N-gram 从 LM 中移除。

4. 二值化

可以使用 bin/build_binary 将 ARPA 格式文件转为二进制格式，查询的时候能显著减少模型加载的时间。KenLM 支持两种类型的二进制数据：

（1）probing，即探测哈希表，n-gram 的 64 位哈希值作为键，对数概率作为值。Probing 方式速度最快，但占用最多内存空间。默认值时 probing。

（2）trie，即前缀树。Trie 方式使用最少内存，但计算 LM 分数略慢一些。

5.2.2.3 训练字级别的 5 元统计语言模型

训练字级别 5-gram LM 的具体流程为：

（1）按 3.2.2.3 小节所述将维基百科中文语料按字进行切分。

（2）按 5.2.2.2 小节所述使用 KenLM 训练 5-gram LM，训练命令为：bin/lmplz -o 5 -S 60% -T /tmp < train.wiki.char.txt > wiki_zh.char.5gram.arpa。

（3）按 5.2.2.2 小节所述使用 bin/build_binary 二值化数据。命令为：bin/build_binary -T /tmp/trie -S 6G trie wiki_zh.char.5gram.arpa wiki_zh.char.5gram.binary.trie，其中指定 /tmp/trie 为临时文件的存放目录，指定排序时至多使用 6GB 内存空间（KenLM 会按需使用），指定二进制数据结构为 trie（桌面工作站内存仅为 16GB）。

最终得到的 ARPA 格式字级别 5-gram 语言模型大小为 19GB，trie 结构的二进制 5-gram 语言模型则为 5.2GB。

5.2.2.4 查询句子的语言模型分数

本文基于 KenLM 的 Python 模块查询句子的 LM 分数，具体步骤为：

（1）基于 KenLM 加载 trie 结构二进制格式的字级别 5-gram 语言模型，得到 model

实例。

（2）将句子按字切分。

（3）通过 model 对象的 score 方法查询句子的 LM 分数。句子的 LM 分数计算如式（5-4）所示。

5.2.2.5 语言模型特征重排序的具体流程

语言模型特征包括 5-gram 语言模型分数和输出句子的 token 数目，LM 重排序的具体流程与 EO 重排序较为类似，如下所示：

（1）启用 n-best 输出机制。同 EO 重排序的第（1）步。

（2）输出格式转换。同 EO 重排序的第（2）步。

（3）获取特征。通过设置-f 参数指定 augmenter.py 脚本使用语言模型特征，并将获取到的两种 LM 特征写入由-o 参数指定的文件，输出文件的格式如图 5-3 所示。

第1个 n-best输出：

0 ||| 系统假设输出1 ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
0 ||| 系统假设输出2 ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
...
0 ||| 系统假设输出*n* ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
1 ||| 系统假设输出1 ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
...
1 ||| 系统假设输出*n* ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
...

第*m*+1个 n-best输出：

m ||| 系统假设输出1 ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数
...
m ||| 系统假设输出*n* ||| F0= NMT解码器分数 LM0= 5-gram LM分数 WordPenalty0= -目标句子的token数目 ||| NMT解码器分数

图 5-3 新增两个 LM 特征的 n-best 输出格式

（4）训练重打分器。同 EO 重排序的第（4）步。

（5）应用重排序机制。同 EO 重排序的第（5）步。

5.2.3 基于编辑操作与语言模型特征的重排序

在前述 EO 重排序（5.2.1.3 小节）和 LM 重排序（5.2.2.5 小节）的基础上，本文将二者进行合并，提出针对中文校对模型 n-best 输出列表的 EO+LM 重排序机制。EO+LM 重排序使用全部的编辑操作和语言模型特征，包括：校对模型解码器分数、三种编辑操作计数特征（删除、插入和替换操作的次数）和两种语言模型特征（5-gram 语言模型分数和校对候选的 token 数目），EO+LM 重排序的具体流程与 EO 或 LM 重排序较为类似，它在第（3）步时通过 augmenter.py 脚本获取三种编辑操作特征和两种语言模型特征并写入文件，其余步骤同 EO 或 LM 重排序。

5.3 基于集成解码和重排序的中文文本自动校对

重排序机制作用于 n-best 输出之上，故其可以与单模型推断或集成解码相结合。重排序方法的应用会使得校对系统的精度下降，而召回率显著提升，故其可以有效提升总体指标。

对单模型推断的 n-best 输出应用重排序，性能提升并不显著，这是因为此时召回率提升对总体指标带来的增益未能大幅超过精确率下降对总体指标带来的损失。由于集成解码能够显著提升系统的精确率，故本小节将前述的集成解码方法（5.1 小节）和重排序组件（5.2 小节）相结合，提出基于集成解码和重排序的中文文本自动校对方法，将重排序组件作用于集成解码的 n-best 输出之上。

基于单模型推断和重排序的中文校对相关实验见 5.4.1 小节，基于集成解码和重排序的中文校对相关实验见 5.4.2 小节。

5.4 实验设计与结果分析

5.4.1 基于单模型推断和重排序的中文文本自动校对相关实验

本小节基于嵌入矩阵不同初始化方式对比实验（表 4-7）报告的实验结果，选用性能最好的单模型——ConvS2S+Char+Word2vec 进行重排序增强对比实验，对比实验结果如表 5-4 所示，报告其在测试集上的性能表现。

表 5-4 基于单模型推断的重排序增强对比实验（基于测试集）

重排序机制	精确率	召回率	M^2 $F_{0.5}$
ConvS2S+Char+Word2vec（single）	**46.50**	12.36	29.95
ConvS2S+Char+Word2vec（single）+EO	37.43	16.17	29.64
ConvS2S+Char+Word2vec（single）+LM	39.51	17.66	31.67
ConvS2S+Char+Word2vec（single）+EO+LM	39.50	**17.68**	**31.68**

对表 5-4 的一些说明：

（1）ConvS2S 表示模型架构为卷积编解码网络；Char 表示字级别的校对模型；Word2vec 表示基于预训练的 word2vec 字向量初始化嵌入矩阵。

（2）（single）表示单模型推断。

（3）EO 表示基于编辑操作特征的重排序方法；LM 表示基于语言模型特征的重排序；EO+LM 表示基于编辑操作和语言模型特征的重排序。

根据表 5-4 的实验结果可以发现：

（1）三种重排序机制均会导致校对系统的精度下降，同时显著提升召回率。

（2）EO 重排序未能提升单模型推断的性能，这是因为此时精确率下降对总体指标带来的损失大于召回率提升对总体指标带来的增益，故需要先应用集成解码再对 n-best 输出进行重打分。

（3）LM 重排序能有效提升总体指标，相较于单模型推断，$F_{0.5}$ 分数提升了 1.72（从 29.95 到 31.67）。

（4）EO+LM 重排序相较于 LM 重排序总体指标几乎没有提升，即 EO 特征的引入并不总是能为 LM 重排序带来正收益。

5.4.2 基于集成解码和重排序的中文文本自动校对相关实验

5.4.1 小节基于单模型推断的重排序增强对比实验表明，对单模型推断的 n-best 输出应用重排序，性能提升并不显著，故本小节进行基于集成解码的重排序增强对比实验，将重排序组件作用于集成解码的 n-best 输出之上。

5.4.2.1 单模型选用 ConvS2S+Char+Word2vec 的相关实验

本小节首先进行 4 模型集成解码，单模型选用 ConvS2S+Char+Word2vec（依据表 4-7 报告的实验结果，其为性能最好的单模型），随机数种子分别设为 1、2、3、4、5、6、1000、2000 和 3000，基于 TITAN V 显卡训练各单模型。各单模型和集成解码在测试集上的性能表现如表 5-5 所示。

表 5-5 基于 ConvS2S+Char+Word2vec 的集成解码对比实验结果

随机数种子	精确率	召回率	M^2 $F_{0.5}$
基于 TITAN V 训练的各单模型			
seed=1	**46.50**	**12.36**	**29.95**
seed=2	42.81	11.23	27.40
seed=3	45.57	11.91	29.11
seed=4	43.54	10.92	27.25
seed=5	44.84	11.07	27.84
seed=6	45.02	11.54	28.49
seed=1000	46.01	12.11	29.49
seed=2000	45.03	11.33	28.23
seed=3000	42.41	11.58	27.68
不同方式的集成解码			
（1，2，3，4）	49.63	10.76	28.81
（1,1000,2000,3000）	**50.56**	10.76	29.07

续表 5-5

（1，3，6，1000）	50.36	**11.24**	**29.69**

由表 5-5 的实验结果可以发现：

（1）集成解码能够显著提升精确率，但召回率会有一定程度下降。

（2）尽管相较于 seed=1 的单模型，不同方式的集成解码均未能取得性能增益，但由于精确率获得了显著提升，故本小节依然基于四模型集成解码进行重排序增强对比实验。

（3）不同随机数种子初始化的单模型最终性能差异较大，性能最好的单模型（seed=1）相较于性能最差的单模型（seed=4）$F_{0.5}$ 分数提升了 2.7 个点（从 27.25 到 29.95）。

本小节接着基于表 5-5 的实验结果基于四模型集成解码进行重排序增强对比实验，对比实验的结果如表 5-6 所示，报告模型在基准测试集上的性能表现。

表 5-6 基于 ConvS2S+Char+Word2vec（4 ens）的重排序增强对比实验（基于测试集）

重排序机制	精确率	召回率	M^2 $F_{0.5}$
ConvS2S+Char+Word2vec（4 ens）	**50.36**	11.24	29.69
ConvS2S+Char+Word2vec（4 ens）+EO	41.30	17.63	32.56
ConvS2S+Char+Word2vec（4 ens）+LM	44.34	17.07	**33.61**
ConvS2S+Char+Word2vec（4 ens）+EO+LM	41.50	**19.08**	33.60

对表 5-6 的一些说明：

（1）Char 表示字级别的校对模型。

（2）（4 ens）表示 4 模型集成解码。

根据表 5-6 的实验结果可以发现：

（1）相较于 4 模型集成解码，三种重排序机制均能显著提升模型性能，LM 重排序优于 EO 重排序，EO+LM 重排序与 LM 重排序总体指标几乎相同。

（2）ConvS2S+Char+Word2vec（4 ens）+LM 优于 ConvS2S+Char+Word2vec（single）+LM，即针对集成解码的 n-best 输出应用重排序的效果优于基于单模型推断（$F_{0.5}$ 分数提升了 1.94，从 31.67 到 33.61），这验证了先进行集成解码再应用重排序是有必要的。

（3）相较于 4 模型集成解码，LM 重排序增强的校对模型获得了约 13.2%的相对性能提升（绝对提升 3.92 个点，从 29.69 到 33.61）。

5.4.2.2 单模型选用 ConvS2S+Char+Random 的相关实验

由于相较于 seed=1 的单模型，基于 ConvS2S+Char+Word2vec 的 4 模型集成解码未能提升 $F_{0.5}$ 分数，故本小节对单模型架构的选择进行进一步的探索。基于嵌入矩阵不同初始化方式对比实验（表 4-7）报告的实验结果，考虑到嵌入矩阵随机初始化的校对模型性能表现亦较好，故本小节的单模型选用 ConvS2S+Char+Random（Random 表示随机初始化嵌入矩阵）。各单模型和集成解码在测试集上的性能表现如表 5-7 所示。

表 5-7 基于 ConvS2S+Char+Random 的集成解码对比实验结果

随机数种子	精确率	召回率	M^2 $F_{0.5}$ 分数
基于 TITAN V 训练的四个单模型和集成解码			
seed=1	44.38	**11.75**	28.53
seed=2	45.15	11.39	28.34
seed=3	45.02	11.29	28.19
seed=4	44.97	11.41	28.31
（1，2，3，4）	**50.31**	10.77	**29.02**
基于 RTX 2080Ti 训练的四个单模型和集成解码			
seed=1	43.56	11.59	28.07
seed=1000	45.95	**11.75**	**29.04**
seed=2000	44.05	11.40	28.01
seed=3000	45.25	11.27	28.23
（1,1000,2000,3000）	**49.26**	10.59	28.47
基于四个指标最高的单模型的集成解码			
（1，2，4，1000）	**51.31**	10.96	**29.55**

由表 5-7 的实验结果可以发现：

（1）集成解码能够显著提升精确率（但召回率会有一定程度下降），进而提升 $F_{0.5}$ 分数。

（2）相较于 seed=1 的单模型，“（1，2，3，4）”4 模型集成解码获得了 0.49 的性能提升（从 28.53 到 29.02），对随机数种子分别设为 1、2、3、4 的四个单模型进行集成解码也是惯例的做法。

（3）相较于 seed=1000 的单模型，“（1，1000，2000，3000）”4 模型集成解码未能取得性能增益，这是因为 seed=1、2000、3000 三个模型性能较差，集成解码反倒拖累了 seed=1000 单模型的性能。

（4）相较于 seed=1 的单模型，基于“（1，2，4，1000）”四个指标最高的单模型的集成解码获得了 1.02 的性能提升（从 28.53 到 29.55），这表明集成解码能有效提升校对模型的性能。

本小节接着基于表 5-7 的实验结果基于四模型集成解码进行重排序增强对比实验，对比实验的结果如表 5-8 所示，报告模型在基准测试集上的性能表现。

表 5-8 基于 ConvS2S+Char+Random（4 ens）的重排序增强对比实验（基于测试集）

重排序机制	精确率	召回率	M^2 $F_{0.5}$
ConvS2S+Char+Random（4 ens）	**51.31**	10.96	29.55
ConvS2S+Char+Random（4 ens）+EO	39.76	**18.73**	32.47
ConvS2S+Char+Random（4 ens）+LM	44.72	17.52	34.12
ConvS2S+Char+Random（4 ens）+EO+LM	44.48	18.06	**34.41**

根据表 5-8 的实验结果可以发现：

（1）EO+LM 重排序略优于 LM 重排序，LM 重排序显著优于 EO 重排序。

（2）相较于 ConvS2S+Char+Word2vec（4 ens）+EO+LM（见表 5-6），ConvS2S+Char+Random（4 ens）+EO+LM 又获得了进一步的性能提升，$F_{0.5}$ 分数绝对提升 0.81 个点（从 33.60 到 34.41）。

（3）相较于 4 模型集成解码，EO+LM 重排序增强的中文校对模型获得了约 16.5% 的相对性能提升（绝对提升 4.86 个点，从 29.55 到 34.41）。

5.4.3 不同中文校对系统在基准测试集上的性能对比

本文还同时报告了其他中文校对系统在 NLPCC 2018 基准测试集上的性能表现[47]，如表 5-9 所示。

表 5-9 不同中文校对系统性能报告（基于测试集）

系统名称	精确率	召回率	M^2 $F_{0.5}$
之前的一些中文校对系统			
BLCU（4 ens）	**47.63**	12.56	30.57
Youdao[27]	35.24	18.64	29.91
AliGM[26]	41.00	13.75	29.36
BLCU[28]	**41.73**	13.08	29.02
PKU	41.22	7.18	21.16

续表 5-9

本文 5.4.1 和 5.4.2 小节实现的中文校对系统			
ConvS2S+Char+Word2vec（single）+EO+LM	39.50	17.68	31.68
ConvS2S+Char+Word2vec（4 ens）+EO+LM	41.50	**19.08**	33.60
ConvS2S+Char+Random（4 ens）+EO+LM	44.48	18.06	**34.41**

对表 5-9 的一些说明：

（1）Youdao 为网易有道 NLP 团队的 NLPCC 2018 CGEC 共享任务冠军解决方案。

（2）AliGM 为阿里巴巴 NLP 团队的提交结果评分。

（3）BLCU 为北京语言大学团队在评测截止前提交的得分，BLCU（4 ens）为比赛结束后他们继续进行的集成解码实验结果的评分（来自 BLCU 的比赛论文）。

（4）PKU 为北京大学计算语言学研究中心的提交得分。

由表 5-9 可知，本章实现的三个中文校对系统均取得了新的 SOTA 分数，其中 ConvS2S+Char+Random（4 ens）+EO+LM 总体得分最高，它相较于 NLPCC 2018 GEC 共享任务冠军解决方案（系统名称为 Youdao），性能相对提升了约 15%（绝对提升 4.5 个点，从 29.91 到 34.41），相较于之前的 SOTA 结果——BLCU（4 ens）性能相对提升了约 12.6%，（绝对提升 3.84 个点，从 30.57 到 34.41）。

5.5 本章小结

本章基于集成解码和重排序增强中文校对模型的性能，阐述了集成解码和三种重排序方法，实验结果表明，集成解码与重排序的组合是有必要的，且三种重排序方法均能显著提升模型性能。

第 6 章 基于多通道融合与重排序的中文文本自动校对

字级别的模型擅长于校对错别字错误，而词级别的模型则更擅长于校对词语搭配错误，多通道融合方法旨在结合不同切分粒度的校对模型，以期能够利用不同级别的信息，避免单独应用字或 BPE 级别模型所带来的局限性。基于多通道融合与重排序的中文文本自动校对通过三个预测通道有效结合不同层次的信息，通过标准化的 LM 特征重排序组件对聚合的多通道输出结果进行重打分，选择得分最高的句子作为系统输出。

本章结合第 4 章和 5.2 小节的工作，将不同粒度的校对模型、集成解码、n-best 输出和重排序进行合并，提出一种新的多通道融合框架。6.1 小节阐述多通道融合与重排序框架，6.2 小节用于确定不同级别的校对模型，6.3 小节为相关对比实验分析。

6.1 多通道融合与重排序架构

多通道融合与重排序架构由三个组件构成：（1）多通道预测组件用于将源端输入翻译为校对候选输出，各通道均启用集成解码和 n-best 输出；（2）聚合组件用于合并各个通道的输出；（3）重排序组件基于标准化的 LM 特征重排序对聚合的多通道输出重打分，选择得分最高的句子作为系统输出。多通道融合与重排序架构如图 6-1 所示。

6.1.1 多通道预测组件

多通道预测组件包括 M1、M2 和 M3 三个预测通道，该组件的具体设置如下：

（1）M1 通道仅包含字级别的校对模型（C1），仅利用字级别的信息。

（2）M2 通道将 M1 通道的输出转为 BPE 序列再送入 BPE 级别的校对模型（B1），这种两阶段的校对模式可以有效结合字级别和 BPE 级别模型的优势，即先通过字级别的模型校对错别字错误，再基于 BPE 级别的模型校对词语搭配错误等。

（3）M3 通道仅包含 BPE 级别的模型（B2），仅利用词级别的信息。

（4）句子送入不同级别的校对模型时需进行相应级别的切分，例如 C1 的输出经过“1-best 输出模块”和“字转 BPE 模块”再送入 B1 中。

（5）三个预测通道均启用集成解码，这能显著提升精确率，进而提升模型性能。

（6）C1、B1 和 B2 的输出都为 n-best 列表，这能有效提升校对系统输出的多样性，进而显著提升召回率，从而提升模型性能。

（7）B1 和 B2 共享 4 个不同随机数种子初始化的 BPE 级别校对模型。

6.1.2 聚合组件

聚合组件用于合并各通道的 n-best 输出，具体流程如下所示：

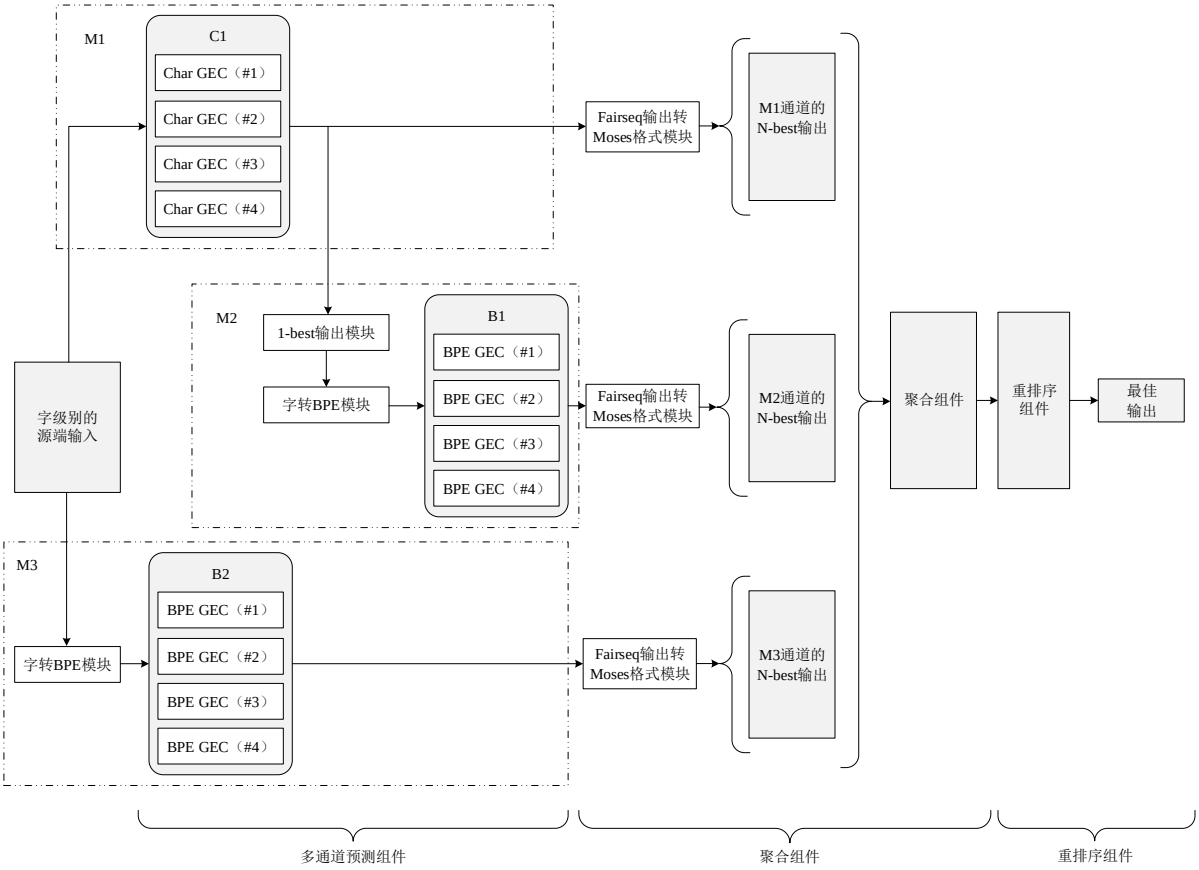

图 6-1 多通道融合与重排序架构图

（1）各通道输出格式转换。根据“O”和“H”标志从 fairseq NLP 库的 interactive.py 的输出中提取源端输入的 n-best 输出及其对应的校对模型解码器分数，将各个通道的输出格式从 fairseq 转为 Moses 格式，Moses 格式如图 5-1 所示。

（2）按照“|||”划分字段，将指定通道的输出按第一个字段进行合并，第一个字段相同意味候选输出对应于同一个源端输入句子。

6.1.3 重排序组件

本章拓展 5.2.2 小节的语言模型特征重排序方法，提出基于标准化语言模型特征的重排序，它将语言模型分数按句子长度标准化，其它同 5.2.2 小节的 LM 特征重排序。

为支持 6.3.2 小节的不同重排序方法对比实验，本章对重排序组件进行相应拓展，拓展后的重排序组件还支持如下 6 种重排序机制：

（1）基于 NMT 解码器分数的重排序，即仅依赖校对模型解码器分数对 n-best 输出重打分。

（2）基于编辑操作特征的重排序，同 5.2.1 小节的 EO 特征重排序。

（3）基于语言模型分数的重排序，即仅依赖外部 5-gram 语言模型计算的句子评分对 n-best 输出重打分，句子评分即语言模型分数，其计算如式（5-4）所示。

（4）基于标准化语言模型分数的重排序，将语言模型分数按句子长度标准化，其它同基于语言模型分数的重排序。

（5）基于语言模型特征的重排序，同 5.2.2 小节的 LM 特征重排序。

（6）基于编辑操作和语言模型特征的重排序，同 5.2.3 小节的 EO+LM 重排序，其中语言模型分数未按句子长度标准化。

本章实现的中文校对系统选用基于标准化语言模型特征的重排序，相关对比实验（见 6.3.2 小节）表明其优于其他重排序机制。

6.2 不同级别的校对模型

6.2.1 字级别的中文校对模型

本文依据 5.4.2 小节重排序对比实验选择 ConvS2S+Char+Random（4 ens）用于 M1 通道的集成解码。根据 5.4.2.2 小节集成解码的实验结果（表 5-7）确定字级别的校对单模型为 ConvS2S+Char+Random（seed=1000），其模型架构为嵌入矩阵随机初始化的字级别卷积编解码网络，随机数种子置为 1000，字级别的校对单模型仅用于 6.3.3 小节的组件拆解对比实验。

6.2.2 子词级别的中文校对模型

根据 BPE 级别模型嵌入矩阵不同初始化方式对比实验的实验结果（表 4-8）可以确定 BPE 级别的校对单模型为 ConvS2S+BPE+Word2vec，其基于预训练的 word2vec BPE 向量初始化嵌入矩阵。

类似于 5.4.2 小节，本小节基于不同的随机数种子训练 BPE 级别的校对单模型，用于在预测通道中进行集成解码。单模型选择 ConvS2S+BPE+Word2vec，随机数种子分别设为 1、2、3、4、5、6，基于 TITAN V 训练各单模型，它们在测试集上的性能表现如表 6-1 所示。

表 6-1 ConvS2S+BPE+Word2vec 不同单模型的性能表现

随机数种子	精确率	召回率	M^2 $F_{0.5}$ 分数
seed=1	42.58	11.61	27.76
seed=2	**43.18**	12.64	**29.12**
seed=3	40.54	12.80	28.28
seed=4	42.06	**12.91**	28.97
seed=5	42.01	12.90	28.95

续表 6-1

seed=6	41.74	12.64	28.58

本小节基于表 6-1 的实验结果基于四模型集成解码进行重排序增强对比实验，对比实验的结果如表 6-2 所示，报告它们在基准测试集上的性能表现。

表 6-2 基于 ConvS2S+BPE+Word2vec（4 ens）的重排序增强对比实验（基于测试集）

重排序机制	精确率	召回率	M^2 $F_{0.5}$
ConvS2S+BPE+Word2vec（4 ens）	**49.39**	11.90	30.31
ConvS2S+BPE+Word2vec（4 ens）+EO 特征+LM 特征	41.56	**19.33**	**33.79**

依据表 6-1 和表 6-2 的实验结果，选择 ConvS2S+BPE+Word2vec（4 ens）用于 M2 和 M3 通道的集成解码，确定 BPE 级别的校对单模型为 ConvS2S+BPE+Word2vec，BPE 级别的校对单模型仅用于 6.3.3 小节的组件拆解对比实验。

6.3 实验设计与结果分析

6.3.1 不同中文校对系统的性能对比

本小节报告不同中文校对系统在 NLPCC 2018 基准测试集上的性能表现，如表 6-3 所示。

表 6-3 不同中文校对系统性能报告（基于测试集）

系统名称	精确率	召回率	M^2 $F_{0.5}$
之前的一些中文校对系统			
BLCU（4 ens）	47.63	12.56	30.57
Youdao[27]	35.24	**18.64**	29.91
AliGM[26]	41.00	13.75	29.36
BLCU[28]	41.73	13.08	29.02
PKU	41.22	7.18	21.16
第 5 章实现的性能最好的中文校对系统			
ConvS2S+Char+Random（4 ens）+EO 特征+LM 特征	44.48	18.06	34.41
第 6 章实现的中文校对系统			
M1+M2+M3（4 ens，n-best）+LM 特征（标准化）	**48.20**	18.51	**36.49**

表 6-3 的实验表明：

（1）本章实现的中文校对系统 M1+M2+M3（4 ens，n-best）+LM 特征（标准化）相较于第五章中最好的中文校对系统性能又获得了进一步的提升，$F_{0.5}$ 分数相对提升约 6.1%（绝对提升 2.08 个点，从 34.41 到 36.49）。

（2）本章实现的中文校对系统相较于 NLPCC 2018 GEC 共享任务冠军解决方案（系统名称为 Youdao），性能相对提升了约 22%（绝对提升 6.58 个点，从 29.91 到 36.49），相较于之前的 SOTA 结果——BLCU（4 ens）性能相对提升了约 19.4%，（绝对提升 5.92 个点，从 30.57 到 36.49）。

6.3.2 不同重排序方法对比实验

本小节进行不同重排序方法对比实验，三个预测通道均启用集成解码，M1 通道的设置按 6.2.1 小节所述选用 ConvS2S+Char+Random（4 ens），M2 和 M3 通道的设置按 6.2.2 小节所述选用 ConvS2S+BPE+Word2vec（4 ens），各通道均输出 N 个最佳候选。不同重排序方法对比实验结果如表 6-4 所示，报告它们在基准测试集上的性能表现。

表 6-4 不同重排序方法对比实验

序号	模型架构	精确率	召回率	M^2 $F_{0.5}$
1	M1+M2+M3（4 ens，n-best）+**LM 分数**	26.35	23.69	25.77
2	M1+M2+M3（4 ens，n-best）+LM 分数（**标准化**）	28.34	**26.17**	27.88
3	M1+M2+M3（4 ens，n-best）+**NMT 分数**	**52.64**	11.91	31.26
4	M1+M2+M3（4 ens，n-best）+**EO 特征**	40.19	20.38	33.65
5	M1+M2+M3（4 ens，n-best）+**EO 特征+LM 特征**	41.46	21.46	34.94
6	M1+M2+M3（4 ens，n-best）+**LM 特征**	45.31	19.24	35.65
7	M1+M2+M3（4 ens，n-best）+LM 特征（**标准化**）	48.20	18.51	**36.49**

对表 6-4 的一些说明：LM 分数表示基于语言模型分数的重排序，LM 分数（标准化）表示基于标准化语言模型分数的重排序，NMT 分数表示基于 NMT 解码器分数的重排序，EO 特征表示基于编辑操作特征的重排序，EO 特征+LM 特征表示基于编辑操作和语言模型特征的重排序，LM 特征表示基于语言模型特征的重排序，LM 特征（标准化）表示基于标准化语言模型特征的重排序。

根据表 6-4 的实验结果可以发现：

（1）LM 特征（标准化）重排序优于其他重排序机制，相较于 LM 特征重排序，LM 特征（标准化）重排序能进一步提升性能，这说明了本文提出的 LM 特征（标准化）重排序方法的有效性。

（2）1 号与 2 号架构的对比，以及 6 号与 7 号架构的对比均表明，将语言模型分数按句子长度标准化能为重排序方法带来正收益。

（3）由于各通道均输出 N 个最佳候选，故各通道聚合后的 n-best 输出可能包含较多误校对，于是仅依赖于 LM 分数的重排序（1 号与 2 号架构）效果较差。

6.3.3 组件拆解对比实验

本小节通过组件拆解对比实验说明各通道启用集成解码和 n-best 输出的必要性，验证集成解码和 n-best 输出必要性的实验结果如表 6-5 所示。

表 6-5 集成解码和 n-best 输出必要性的验证实验

模型架构	精确率	召回率	M^2 $F_{0.5}$
M1+M2+M3（**single**，n-best）+LM 特征（标准化）	41.12	**19.27**	33.52
M1+M2+M3（4 ens，**1-best**）+LM 特征（标准化）	**49.35**	14.04	32.84
M1+M2+M3（4 ens，n-best）+LM 特征（标准化）	48.20	18.51	**36.49**

对表 6-5 的一些说明：single 表示各通道均为单模型推断，1-best 表示各通道均只输出最佳校对候选。

表 6-5 的实验结果表明：

（1）各通道均只进行单模型推断会导致精确率大幅下降，进而导致 $F_{0.5}$ 分数显著降低，这表明各通道均启用集成解码能显著提升精确率，对于多通道融合与重排序框架而言是非常有必要的。

（2）各通道均只输出最佳校对候选会导致召回率显著下降，进而导致 $F_{0.5}$ 分数显著降低，这表明各通道均启用 n-best 输出能有效提升校对系统输出的多样性，进而显著提升召回率，对于多通道融合与重排序框架是非常有必要的。

本小节还通过通道拆解对比实验说明，M1、M2 和 M3 这三个预测通道对于本章提出的多通道融合与重排序中文文本校对方法而言都是有必要的，实验结果如表 6-6 所示。

表 6-6 通道拆解对比实验

模型架构	精确率	召回率	M^2 $F_{0.5}$
M1+M2+M3（4 ens，n-best）+ LM 特征（标准化）	48.20	18.51	**36.49**
M1+M2（4 ens，n-best）+ LM 特征（标准化）	46.90	**19.04**	36.28
M1（4 ens，n-best）+ LM 特征（标准化）	46.34	15.96	33.57

表 6-6 的实验结果表明，M1、M2 和 M3 三个预测通道都是有必要的，M1+M2 相较于仅使用 M1 召回率和 $F_{0.5}$ 分数均显著提升，这说明 M2 通道的两阶段校对方法是有效的。

6.4 本章小结

本章基于多通道融合与重排序进一步提升中文校对模型的性能，阐述了多通道融合与重排序架构，确定了不同级别的校对模型，相关对比实验表明了基于多通道融合与重排序的中文校对方法的有效性，组件拆解实验表明各组件都是有必要的。

结论与展望

本文工作总结

随着 GEC 共享任务的举办、公开训练语料和基准测试集的发布，文本自动校对任务受到越来越多研究人员的关注。目前深度学习在 NLP 领域得到了广泛应用，机器翻译任务亦取得了一系列突破性进展，这些都促使研究人员将 seq2seq NMT 模型应用到 GEC 任务中。本文主要的工作及贡献如下：

（1）预处理训练数据和实现标准性能评估方法。预处理 NLPCC 2018 GEC 共享任务训练集，将其转为校对平行语料，用于训练中文校对模型。预处理维基百科中文语料，切分后的文本用于训练词向量和 N-gram 语言模型。对校对系统输出的序列用官方分词工具进行重切分，再使用官方性能评估脚本计算校对任务标准评估指标。

（2）针对校对任务的性质，提出一种基于字级别卷积编解码网络的中文校对模型，并通过扩展训练平行语料、移除异常句子对和使用预训练的字向量初始化嵌入矩阵进一步提升模型性能。数据扩增对比实验表明，使用更多高质量的校对平行语料能显著增强模型性能。数据清洗对比实验表明，过滤目标端长度远小于源端的句子对能有效提升性能。不同级别建模对比实验表明字级别的切分粒度优于词级别和子词级别。嵌入矩阵不同方式初始化的对比实验表明，使用预训练的 word2vec 字向量能够有效提升字级别模型的性能。

（3）提出一种基于集成解码和重排序的中文文本自动校对方法，它针对 GEC 模型集成解码的 N 个最佳输出应用重排序机制，通过结合 GEC 模型解码器分数、针对校对任务的编辑操作特征和五元语言模型分数，对输入错句对应的 N 个最佳候选输出重打分，得分最高的句子即为输入错句的最佳纠正输出。相关实验结果表明，集成解码与重排序的组合是有必要的，且重排序机制能显著提升模型性能。

（4）提出一种基于多通道融合与重排序的中文文本自动校对方法。多通道融合与重排序架构结合字级别和子词级别的校对模型，通过三个预测通道有效结合不同层次的信息，各预测通道均启用集成解码并输出 N 个最佳候选，最后再对聚合的多通道输出结果应用标准化的 LM 特征重排序打分并选出最佳输出。实验结果表明了基于多通道融合与重排序的中文校对方法的有效性，组件拆解实验表明各组件都是有必要的。

未来工作展望

未来工作：

（1）可以尝试使用其他的 seq2seq 模型架构构建中文校对基线模型，例如 LSTM seq2seq 和 Transformer 模型等。

（2）构建更大规模的训练集。本文 4.2.1 小节实验表明增加高质量的平行语料能显著提升模型性能。

（3）应用数据增强方法。Edunov 等[79]提出的反向翻译技术已成为当前机器翻译的基准方法，Ge 等[24]则提出三种针对英文 GEC 任务的流畅度提升学习方法。

（4）本文基于 N-gram 语言模型计算语言模型分数，可以尝试使用 RNNLM[50]、AWD-LSTM[80]或 AWD-LSTM-MoS[81]等神经网络语言模型改进 LM 相关的重排序组件。

（5）本文的多通道融合与重排序架构仅使用了 NMT 组件，可以尝试融合基于分类器的 GEC 组件和基于 SMT 的组件。

致 谢

文笔至此，转眼间我在交大的求学生涯就快要画上句号了。回想这三年的读研生涯，多少往事涌上心头，三年的求学探索，我经历了磨练，也得到了成长。

感谢我的研究生导师李天瑞教授！李老师处事态度认真严谨，而跟随李老师进行学习，更让我感受到了李老师高超的学术造诣和严谨治学的学术精神。李老师不仅学术水平令人敬佩，在体育竞技、为人处世等方面更散发着独特的魅力。李老师不仅是我学术上的导师，更是我生活的榜样。感谢您一直以来对我的辛勤指导，您的谆谆教诲将使我受益终身！同时我还要感谢贾真老师，感谢这三年贾老师对我学习给予的帮助。

感谢我的父母！从小学开始，我便远离家乡外出求学，到现在已有近二十年光景。当初稚嫩无知的我，虽然没有你们在身边的陪伴，但耳边总有你们嘘寒问暖的声音，感谢你们一直以来对我的关爱和鼓励。

感谢张涛学长、李金亮学长、唐敏学长和曹宇同学，我们同在自然语言处理小组学习成长，与你们的讨论让我获益匪浅，使我的领域能力得到了提升。感谢博士后刘胜久师兄，研究生三年给予了我许多宝贵的学术建议。

感谢我的好朋友叶少晖、邱培熠、赵祥龙、王东杰、周威，在交大能与你们相识，是我人生最宝贵的一笔财富。

感谢实验室的陈超同学、宁尚明同学、易小群同学、朱磊同学、朱立霞同学。三年的时间有你们为伴，给我的学习生涯带来了无数的感动与喜悦。

感谢实验室所有老师，师兄师姐，师弟师妹的帮助和指导意见。

感谢母校西南交通大学七年来对我的栽培，感谢四川省云计算与智能技术高校重点实验室为我提供了良好的实验及学习环境。

最后，感谢百忙之中抽空对我的论文进行审阅的各位老师，感谢你们提出的宝贵意见！

参考文献

[1] Chollampatt S, Ng H T. A multilayer convolutional encoder-decoder neural network for grammatical error correction[C]//Proceedings of the 32nd AAAI Conference on Artificial Intelligence. 2018.

[2] Zheng B, Che W, Guo J, et al. Chinese grammatical error diagnosis with long short-term memory networks[C]//Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016). 2016: 49-56.

[3] Dahlmeier D, Ng H T, Ng E J F. NUS at the HOO 2012 Shared Task[C]//Proceedings of the 7th Workshop on Building Educational Applications Using NLP. 2012: 216-224.

[4] Rozovskaya A, Chang K W, Sammons M, et al. The Illinois-Columbia system in the CoNLL-2014 shared task[C]//Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task. 2014: 34-42.

[5] Rozovskaya A, Roth D. Grammatical error correction: Machine translation and classifiers[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016, 1: 2205-2215.

[6] Junczys-Dowmunt M, Grundkiewicz R. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction[J]. arXiv preprint arXiv:1605.06353, 2016.

[7] Chollampatt S, Ng H T. Connecting the dots: Towards human-level grammatical error correction[C]//Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications. 2017: 327-333.

[8] 徐连诚, 石磊. 自动文字校对动态规划算法的设计与实现[J]. 计算机科学, 2002, 29(9): 149-150.

[9] 龚小谨, 罗振声, 骆卫华. 中文文本自动校对中的语法错误检查[J]. 计算机工程与应用, 2003, 39(8): 98-100.

[10] 陈笑蓉, 秦进, 汪维家, 等. 中文文本校对技术的研究与实现[J]. 计算机科学, 2003, 30(11): 53-55.

[11] 刘亮亮, 曹存根. 基于局部上下文特征的组合的中文真词错误自动校对研究[J]. 计算机科学, 2016(12): 37-42.

[12] 刘亮亮, 曹存根. 中文“非多字词错误”自动校对方法研究[J]. 计算机科学, 2016, 43(10): 200-205.

[13] 张涛. 中文文本自动校对系统设计与实现[D]. 西南交通大学, 2017.

[14] 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, 2017, 40(4): 911-924.

[15] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv:1406.1078, 2014.

[16] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//Advances in Neural Information Processing Systems. 2014: 3104-3112.

[17] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[18] Luong M T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation[J]. arXiv preprint arXiv:1508.04025, 2015.

[19] Gehring J, Auli M, Grangier D, et al. Convolutional sequence to sequence learning[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 1243-1252.

[20] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

[21] Yuan Z, Briscoe T. Grammatical error correction using neural machine translation[C]//Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2016: 380-386.

[22] Xie Z, Avati A, Arivazhagan N, et al. Neural language correction with character-based attention[J]. arXiv preprint arXiv:1603.09727, 2016.

[23] Ji J, Wang Q, Toutanova K, et al. A nested attention neural hybrid model for grammatical error correction[J]. arXiv preprint arXiv:1707.02026, 2017.

[24] Ge T, Wei F, Zhou M. Fluency boost learning and inference for neural grammatical error correction[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018, 1: 1055-1065.

[25] Lichtarge J, Alberti C, Kumar S, et al. Weakly Supervised Grammatical Error Correction using Iterative Decoding[J]. arXiv preprint arXiv:1811.01710, 2018.

[26] Zhou J, Li C, Liu H, et al. Chinese grammatical error correction using statistical and neural models[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 117-128.

[27] Fu K, Huang J, Duan Y. Youdao's Winning solution to the NLPCC-2018 Task 2 challenge: a neural machine translation approach to Chinese grammatical error correction[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 341-350.

[28] Ren H, Yang L, Xun E. A Sequence to Sequence Learning for Chinese Grammatical Error Correction[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2018: 401-410.

[29] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[30] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[31] Ling W, Dyer C, Black A W, et al. Two/too simple adaptations of word2vec for syntax problems[C]//Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015: 1299-1304.

[32] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space[J]. arXiv preprint arXiv:1301.3781, 2013.

[33] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality[C]//Advances in Neural Information Processing Systems. 2013: 3111-3119.

[34] Och F J. Minimum error rate training in statistical machine translation[C]//Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003: 160-167.

[35] Ng H T, Wu S M, Wu Y, Hadiwinoto C, Tetreault J. (2013). The CoNLL-2013 Shared Task on Grammatical Error Correction.[C]//Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task. 2013: 1-12.

[36] Ng H T, Wu S M, Briscoe T, et al. The CoNLL-2014 shared task on grammatical error correction[C]//Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task. 2014: 1-14.

[37] Nicholls D. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT[C]//Proceedings of the Corpus Linguistics 2003 Conference. 2003, 16: 572-581.

[38] Luong M T, Manning C D. Achieving open vocabulary neural machine translation with hybrid word-character models[J]. arXiv preprint arXiv:1604.00788, 2016.

[39] Bojanowski P, Grave E, Joulin A, et al. Enriching word vectors with subword information[J]. Transactions of the Association for Computational Linguistics, 2017, 5: 135-146.

[40] Sennrich R, Haddow B, Birch A. Neural machine translation of rare words with subword units[J]. arXiv preprint arXiv:1508.07909, 2015.

[41] Tajiri T, Komachi M, Matsumoto Y. Tense and aspect error correction for ESL learners using global context[C]//Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012: 198-202.

[42] Dahlmeier D, Ng H T, Wu S M. Building a large annotated corpus of learner English: The NUS corpus of learner English[C]//Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications. 2013: 22-31.

[43] 张仰森, 曹元大, 俞士汶. 基于规则与统计相结合的中文文本自动查错模型与算法[J]. 中文信息学报, 2006, 20(4): 3-9, 57.

[44] 王燚. 基于 Trie 结构的带通配符的相似字符串匹配算法[J]. 计算机应用, 2004, 24(10): 121-124.

[45] Gaoqi R A O, Zhang B, Endong X U N, et al. IJCNLP-2017 task 1: Chinese grammatical error diagnosis[C]//Proceedings of the International Joint Conference on Natural Language Processing 2017, Shared Tasks. 2017: 1-8.

[46] Xie P. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task[C]//Proceedings of the International Joint Conference on Natural Language Processing 2017, Shared Tasks. 2017: 41-46.

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

[48] Ott M, Edunov S, Baevski A, et al. fairseq: A Fast, Extensible Toolkit for Sequence Modeling[J]. arXiv preprint arXiv:1904.01038, 2019.

[49] Cao S, Lu W, Zhou J, et al. cw2vec: Learning chinese word embeddings with stroke n-gram information[C]//Proceedings of the 32nd AAAI Conference on Artificial Intelligence. 2018.

[50] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model[C]//Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010. 2. 1045-1048.

[51] Dauphin Y N, Fan A, Auli M, et al. Language modeling with gated convolutional networks[C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 933-941.

[52] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

[53] Wu Y, Schuster M, Chen Z, et al. Google's neural machine translation system: Bridging the gap between human and machine translation[J]. arXiv preprint arXiv:1609.08144, 2016.

[54] Gu J, Bradbury J, Xiong C, et al. Non-autoregressive neural machine translation[J]. arXiv preprint arXiv:1711.02281, 2017.

[55] Guo J, Tan X, He D, et al. Non-autoregressive neural machine translation with enhanced decoder input[J]. arXiv preprint arXiv:1812.09664, 2018.

[56] Wang Y, Tian F, He D, et al. Non-Autoregressive Machine Translation with Auxiliary Regularization[J]. arXiv preprint arXiv:1902.10245, 2019.

[57] Sammut C. Encyclopedia of Machine Learning[M]. 1st ed. Springer, 2010.

[58] Schuster M, Nakajima K. Japanese and korean voice search[C]//2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012: 5149-5152.

[59] Harris Z S. Distributional structure[J]. Word, 1954, 10(2-3): 146-162.

[60] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model[J]. Journal of Machine Learning Research, 2003, 3(Feb): 1137-1155.

[61] Wieting J, Bansal M, Gimpel K, et al. Charagram: Embedding words and sentences via character n-grams[J]. arXiv preprint arXiv:1607.02789, 2016.

[62] Chen X, Xu L, Liu Z, et al. Joint learning of character and word embeddings[C]// Proceedings of the 24th International Joint Conference on Artificial Intelligence. 2015.

[63] Sun Y, Lin L, Yang N, et al. Radical-enhanced chinese character embedding[C]//International Conference on Neural Information Processing. Springer, Cham, 2014: 279-286.

[64] Li Y, Li W, Sun F, et al. Component-enhanced chinese character embeddings[J]. arXiv preprint arXiv:1508.06669, 2015.

[65] Yu J, Jian X, Xin H, et al. Joint embeddings of chinese words, characters, and fine-grained subcharacter components[C]//Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. 2017: 286-291.

[66] Micikevicius P, Narang S, Alben J, et al. Mixed Precision Training[J]. 2017.

[67] Rehurek R, Sojka P. Software framework for topic modelling with large corpora[C]//In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. 2010.

[68] 北京语言大学语言资源高精尖创新中心. HSK 动态作文语料库 2.0 版[EB/OL]. http://yuyanziyuan.blcu.edu.cn/art/2018/4/10/art_12635_1129752.html, 2019-4-9.

[69] Dahlmeier D , Ng H T . Better evaluation for grammatical error correction[C]//

Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2012.

[70] Khandelwal U, He H, Qi P, et al. Sharp nearby, fuzzy far away: How neural language models use context[J]. arXiv preprint arXiv:1805.04623, 2018.

[71] Nesterov Y. A method for unconstrained convex minimization problem with the rate of convergence O (1/k^ 2)[C]//Doklady AN USSR. 1983, 269: 543-547.

[72] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[73] Sennrich R, Haddow B, Birch A. Improving neural machine translation models with monolingual data[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics Companion. 2016: 86-96.

[74] Jozefowicz R, Vinyals O, Schuster M, et al. Exploring the limits of language modeling[J]. arXiv preprint arXiv:1602.02410, 2016.

[75] Junczys-Dowmunt M, Grundkiewicz R. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation[C]//Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task. 2014: 25-33.

[76] Koehn P, Hoang H, Birch A, et al. Moses: Open source toolkit for statistical machine translation[C]//Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion. 2007: 177-180.

[77] Heafield K. KenLM: Faster and smaller language model queries[C]//Proceedings of the 6th Workshop on Statistical Machine Translation. Association for Computational Linguistics, 2011: 187-197.

[78] Chen S F, Goodman J. An empirical study of smoothing techniques for language modeling[J]. Computer Speech & Language, 1999, 13(4): 359-394.

[79] Edunov S, Ott M, Auli M, et al. Understanding back-translation at scale[J]. arXiv preprint arXiv:1808.09381, 2018.

[80] Merity S, Keskar N S, Socher R. Regularizing and optimizing LSTM language models[J]. arXiv preprint arXiv:1708.02182, 2017.

[81] Yang Z, Dai Z, Salakhutdinov R, et al. Breaking the softmax bottleneck: A high-rank RNN language model[J]. arXiv preprint arXiv:1711.03953, 2017.

攻读硕士学位期间发表的论文及科研成果

已录用的论文

[1] 杨宗霖, 李天瑞, 刘胜久, 等. 基于 Spark Streaming 的流式并行文本校对[J]. 计算机科学, 2019.（已录用）

参与的科研项目

[1] 国家自然科学基金面上项目: 面向城市大数据的深度学习模型与方法研究（项目编号：61773324）. 起止时间: 2018/01-2021/12.

[2] 四川省科技服务业示范项目: 智能审校云服务平台开发与应用示范（项目编号：2016GFW0167）. 起止时间: 2016/01-2017/12.