UMRSpell: Unifying the Detection and Correction Parts of Pre-trained Models towards Chinese Missing, Redundant, and Spelling Correction

Zheyu He[1,†], Yujin Zhu[1,†], Linlin Wang[2,3], Liang Xu[1,*]
[1]GammaLab, PingAn OneConnect, Shanghai, China
[2]East China Normal University
[3]Shanghai Artificial Intelligence Laboratory
hezheyu874@pingan.com.cn, zhuyujin204@ocft.com,
llwang@cs.ecnu.edu.cn., xlpaul@126.com

Abstract

Chinese Spelling Correction (CSC) is the task of detecting and correcting misspelled characters in Chinese texts. As an important step for various downstream tasks, CSC confronts two challenges: 1) Character-level errors consist not only of spelling errors but also of missing and redundant ones that cause variable length between input and output texts, for which most CSC methods could not handle well because of the consistence length of texts required by their inherent detection-correction framework. Consequently, the two errors are considered outside the scope and left to future work, despite the fact that they are widely found and bound to CSC task in Chinese industrial scenario, such as Automatic Speech Recognition (ASR) and Optical Character Recognition (OCR). 2) Most existing CSC methods focus on either detector or corrector and train different models for each one, respectively, leading to insufficiency of parameters sharing. To address these issues, we propose a novel model **UMRSpell** to learn detection and correction parts together at the same time from a multi-task learning perspective by using a detection transmission self-attention matrix, and flexibly deal with both missing, redundant, and spelling errors through re-tagging rules. Furthermore, we build a new dataset **ECMR-2023** containing five kinds of character-level errors to enrich the CSC task closer to real-world applications. Experiments on both SIGHAN benchmarks and ECMR-2023 demonstrate the significant effectiveness of UMRSpell over previous representative baselines.

1 Introduction

Recent decades have witnessed the comprehensive development of one important Neural Language Processing (NLP) task: Chinese Spelling Correction (CSC), which focus on detecting and correcting spelling errors in texts (Yu and Li, 2014). The task shares a long research line originated from early 1990s (Shih et al., 1992; Chang, 1995), but remains challenging in Chinese context because that different from English, many Chinese characters are phonologically and visually similar while semantically diverse (Liu et al., 2010). Moreover, CSC not only plays an essential role for various downstream tasks such as search engine (Martins and Silva, 2004; Gao et al., 2010) or automatic essay scoring (Burstein and Chodorow, 1999), but also becomes a necessary part in some widely used industrial applications such as Automatic Speech Recognition (ASR) (Sarma and Palmer, 2004; Errattahi et al., 2018) and Optical Character Recognition (OCR) (Afli et al., 2016; Hládek et al., 2020) systems.

Early work on CSC includes but not limited to pipeline strategy (Chen et al., 2013), traditional language model (Yu and Li, 2014), and sequence-to-sequence learning (Wang et al., 2019). With the growth of the deep learning techniques and the insight for the task itself, the subsequent studies gradually summarize and focus on two crucial parts of CSC, defined as detection part and correction part. Generally, a CSC system derives representations (pronunciation, pinyin, glyph, shape, strokes, and so on) for characters from audio and visual modalities to locate the misspelled ones in its detection part, and then output the text without spelling errors in its correction part (Zhang et al., 2020; Cheng et al., 2020; Bao et al., 2020; Hong et al., 2019). Currently, since the pre-trained masked language models with attention mechanism (Vaswani et al., 2017; Devlin et al., 2019) achieve impressive performance in many NLP domains, they are also introduced into CSC (Zhu et al., 2022; Liu et al., 2022, 2021).

Despite its development, there are still two weaknesses in CSC that bring bottleneck in performance: Firstly, in real world scenario, especially for ASR and OCR tasks, there is not only misspelled case

[†]Equal contributions.
[*]Corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10238–10250

but also missing and redundant errors happening in character-level preprocessing. Missing characters mean a lack of characters needed to be inserted into the identified position, while redundant characters mean useless or repeated characters needed to be deleted (Zheng et al., 2021). Most existing CSC literatures (Zhang et al., 2021, 2020; Cheng et al., 2020; Zhu et al., 2022; Bao et al., 2020) consider the issue belonging to another similar but more complicated task called Chinese Grammatical Error Correction (CGEC) (Zheng et al., 2016; Zhang et al., 2022; Wang et al., 2022) and leave them in the future work.[1] The key point is that dealing with such errors might change the length of the input text, while CSC methods based on detection-correction framework could not handle it well due to inconsistency between inputs and outputs of their correction-part (Zheng et al., 2021). Secondly, detection-correction framework-based methods usually concentrate on how to utilize more useful features in either of two parts and might pre-train or fine-tune different language models for each part, respectively, leading to insufficiency of information sharing between detection and correction parts.

In this paper, we proposed a novel pre-trained model, abbreviated as **UMRSpell**, to **U**nify detection and correction parts for Chinese **M**issing, **R**edundant, and **Spell**ing correction. UMRSpell considers each of detection and correction parts as a sub-task and pre-trains both parts based on one backbone together. Inspired by a multi-task idea in multilingual learning (Ouyang et al., 2021), we modify the Back-Translation Masked Language Modeling (BTMLM) objective to obtain mutual information from the concatenated input texts. Moreover, to solve the variable text length issue caused by missing or redundant errors, we apply several methods during the training and predication processes, such as symmetric concatenation, sequence re-tagging rules, and post consistence check. Additionally, an automatically optimized weighted parameter λ is used for the loss calculation, instead of manually selection. Finally, we construct the **E**xtended **C**SC dataset with **M**issing and **R**edundant errors (**ECMR-2023**) in an unsupervised way with the help of a self-selected phonetic and morphological Chinese character similarity dictionary. Our contributions are concluded as follows:

- We propose UMRSpell model to train detection and correction parts together through a special attention mechanism to transmit information of detection part to correction one. Different from most previous work, UMRSpell is adapted to be both detector and corrector in prediction process.
- UMRSpell model is flexible enough to handle not only spelling errors in existing CSC task, but also missing and redundant errors widely found in real world applications.
- We construct ECMR-2023 dataset containing five kinds of character errors, in order to compensate for the lack of missing and redundant samples in the existing CSC tasks.

2 Related Work

Early work on CSC follows the pipeline of error identification, candidate generation and selection, or selects candidates in an unsupervised way from a confusion set. Then methods utilizing traditional language models or considering sequence-to-sequence structure are also proposed (Liu et al., 2010, 2013; Chen et al., 2013; Yu and Li, 2014; Tseng et al., 2015).

Reviewing the studies in recent years, we further summarize CSC methods into seven types shown in Figure 1, mainly depending on which part of detection-correction framework they primarily focus on:

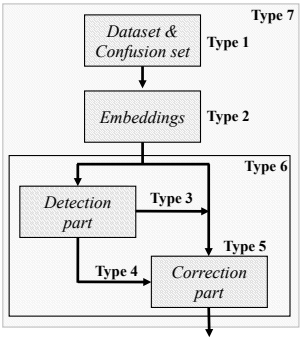

Figure 1: Focal points of seven types of CSC models in detection-correction framework.

Inheriting ideas from earlier work, methods of **Type-1** concern dataset and confusion set. Automatic Corpus Generation proposes a hybrid way to

[1] Nowadays, CGEC task usually contains more kinds of grammatical errors in both character and word levels, such as misused and disordered word, mixing syntax patterns, logical inconsistency, ambiguity, and so on. It is beyond the scope of this paper.

generate labeled spelling errors (Wang et al., 2018). Confusionset-guided Pointer Networks utilizes the confusion set for guiding the character generation (Wang et al., 2019).

Methods of **Type-2** discuss about how to combine embeddings from multi-modal features better for the subsequent detection or correction models. MLM-phonetics is pre-trained to provide embeddings including pronunciation and pinyin of characters (Zhang et al., 2021). PHMOSpell derives and integrates pinyin and glyph representations to a language model by an adaptive gating mechanism (Huang et al., 2021). REALISE captures and selectively mixing the semantic, phonetic and graphic information of characters (Xu et al., 2021).

Methods from both **Type-3** and **Type-4** introduce useful features of characters into general detection process. The difference between them lies in the order in which they guide and affect correction models. Being a typical method of **Type-3**, Soft-Masked BERT (SMBERT) designs a soft masking process after detection part, calculating the weighted sum of the input and [MASK] embeddings weighted by the error probabilities to mask the likely errors in the sequence (Zhang et al., 2020). Alignment-Agnostic Model adds a modification logic unit following detection network to reformulate the sequence to support missing and redundant cases (Zheng et al., 2021).

As a **Type-4** method, SpellGCN builds a graph over the characters and maps it into a set of interdependent detection classifiers that are applied to the representations extracted by BERT (Cheng et al., 2020). Dynamic Connected Networks generates the candidate characters via a pinyin-enhanced generator and model the dependencies between characters through attention mechanism (Wang et al., 2021). MDCSpell captures features of characters and fuses the hidden states of corrector with that of detector to minimize the misleading impact from the misspelled ones (Zhu et al., 2022). The output length of detector from **Type-4** needs to be consistent with that of corrector, while methods of **Type-3** might avoid this issue by adding a unit to re-tag the text from detector before corrector.

Methods of **Type-5** directly handle complicated features in their correction part. Chunk-based Model designed a decoding part and generates all possible chunk candidates for the partially decoded correction (Bao et al., 2020). CRASpell forces correction model to yield similar outputs based on original contexts and constructed noisy ones (Liu et al., 2022).

Methods of **Type-6** introduce prior knowledge and feature information into either detection or correction parts by training large language models. FASPell consists of a denoising auto-encoder and a decoder, and fine-tunes the masked language model in novel ways (Hong et al., 2019). SpellBERT fuses pinyin and radical features through a relational graph convolutional network and train them with a 4-layer BERT (Ji et al., 2021). PLOME is a task-specific language model that jointly learns semantics and misspelled knowledge according to the confusion set based on specially designed masking strategy on BERT (Liu et al., 2021).

Recently, methods focus on the learning strategy, such as Curriculum Learning (Gan et al., 2021) and Adversarial Learning (Li et al., 2021), make progress and are categorized in **Type-7**.

According to above taxonomy, the proposed UMRSpell could be regarded as a combination between **Type-3** and **Type-6**.

3 Methodology

In this section, we first formulate the task, then dive into the structure of UMRSpell, introducing its training and prediction processes. Furthermore, we present how to built ECMR-2023.

3.1 Task Formulation

The Chinese Spelling Correction (CSC) task aims to detect and correct the errors in the Chinese language. When given a text sequence $X = \{x_1, x_2, \ldots, x_n\}$ consisting of n characters, the model takes X as the input and outputs a target character sequence $Y = \{y_1, y_2, ..., y_n\}$ (Cheng et al., 2020; Zhang et al., 2020; Zhu et al., 2022). In this work, CSC is regarded as a multi-task learning that consists of two sequence tagging sub-tasks. In pre-training and fine-tuning, the original text X_{wrong} is concatenated to its masked correct version X_{mask} to form the input whole sequence X for the model. In prediction process, since there is only the input text X_{test} for test, it is copied and concatenates to X_{copy}. It could be found that X_{wrong} and X_{test} are for detection sub-task while X_{mask} and X_{copy} are for correction sub-task. Thereafter, the model outputs both the tagging sequence Y_{tag} of detection part and the text sequence Y_{text} of correction part. Note that the length of Y_{text} could be different from the original text X_{test}.

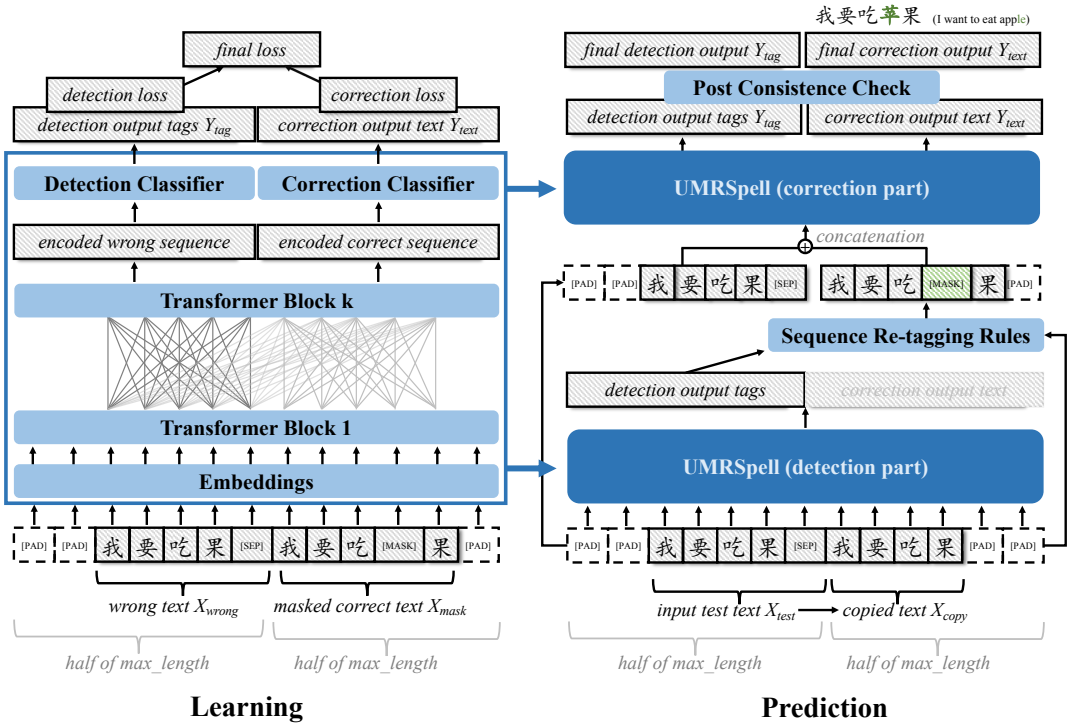

Figure 2: Overview of UMRSpell. Either pre-training or fine-tuning has the same learning process that is given on the left, while prediction process is given on the right. It is seen that UMRSpell could be used in both detection and correction parts during prediction process. Important structures are illustrated in blue squares, while data and loss are drawn in grey squares.

3.2 Structure of UMRSpell

The overview of UMRSpell is illustrated in Figure 2. The structure of the proposed model can be found on the left of the figure as well as the learning process, while the prediction process is demonstrated on the right. Keypoints of improvement are described in the following parts.

Symmetric Concatenation Strategy Before token-, segment-, and position-embeddings, the strategy fills input characters from the middle to the sides, ensuring that the sequences of either detection part or correction part keeps the same length, i.e., half of the maximum input length for the model, in order to make the subsequent attention matrix easy to impart weights of tokens from both X_{wrong} and X_{mask}. Accordingly, the whole input sequence X can be expanded as [[*PAD*], ..., [*PAD*],[*CLS*], X_{wrong}, [*SEP*], X_{mask}, [*PAD*], ..., [*PAD*]] instead of traditional [[*CLS*], X_{wrong}, [*SEP*], X_{mask}, [*PAD*], ..., [*PAD*]]. Correspondingly, the input sequence X in prediction process can be expanded as [[*PAD*], ..., [*PAD*],[*CLS*], X_{test}, [*SEP*], X_{copy}, [*PAD*], ..., [*PAD*]].

Detection Transmission Self-attention Inspired by self-attention matrix used in BTMLM (Ouyang et al., 2021), we adopt the detection transmission self-attention matrix to learn mutual information between two parts, X_{wrong} and X_{mask}, so as to ensure the coherence between detection and correction parts. As shown in Figure 3, the detection-oriented X_{wrong} is attended by itself, while the correction-oriented X_{mask} is attended by both itself and the detection part, implying that information of detection part is transmitted to correction part.

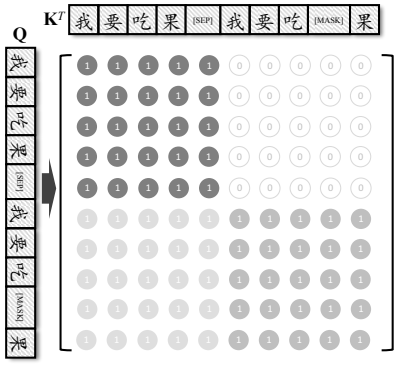

Figure 3: Self-attention mask matrix used in UMRSpell.

Detection and Correction Classifiers Two fully

connected networks are built following encoders to do the sequence labelling classification in token-level for detection part and correction part, respectively. As shown on the left of Figure 2, the encoded tensor is split into two parts with equal length that perform detection and correction independently:

$$P_d(y|X_{e\_wrong}) = \mathrm{Softmax}(\mathrm{W_d h_d + b_d}) \quad (1)$$

$$P_c(y|X_{e\_mask}) = \mathrm{Softmax}(\mathrm{W_c h_c + b_c}) \quad (2)$$

where, $\boldsymbol{W}_d \in \boldsymbol{R}^{n\times 768}$ and $\boldsymbol{W}_c \in \boldsymbol{R}^{n\times 768}$ are weighted matrices, $\boldsymbol{h}_d \in \boldsymbol{R}^{768\times 1}$ and $\boldsymbol{h}_c \in \boldsymbol{R}^{768\times 1}$ are hidden tensors, $\boldsymbol{b}_d \in \boldsymbol{R}^{n\times 1}$ and $\boldsymbol{b}_c \in \boldsymbol{R}^{n\times 1}$ are bias tensors, P_d and P_c are predicted results, for detection and correction parts, respectively. In addition, X_{e_wrong} and X_{e_mask} represent the encoded wrong sequence and masked correct sequence from transformer encoders, respectively. In detector, n represents the number of labels, while in corrector, n represents the number of words in the vocabulary. As a result, the detection classifier outputs the predicted tags Y_{tag} for the input wrong text X_{wrong}, while the correction classifier outputs the predicted text Y_{text} compared with the input masked text X_{mask}.

Sequence Re-tagging Rules During the prediction process, rules are designed to retag the original input text (actually it is the copied version X_{copy}) on consideration to the output tags from UMRSpell (detection part). Following the previous work (Zheng et al., 2021), we give the following re-tagging rules: 1) *misspell*: use [MASK] token to replace the error tokens in X_{copy}, and turn the tag of error tokens from "O" to "S" in X_{test}; 2) *missing error*: insert [MASK] token in the missing places in X_{copy}, and turn the tag of neighbor tokens before and after the missing place from "O" to "M" in X_{test}; 3) *redundant error*: delete the redundant tokens in X_{copy}, and turn the tag of all redundant tokens from "O" to "R" in X_{test}[2].

Post Consistence Check If the final correction output text Y_{text} from UMRSpell (correction part) is the same to the input one, implying that there is no correction happening, but the final detection output tags Y_{tag} reports errors, the detected error tags would be neglected.

[2]"M": missing error, "R": redundant error, "S": misspell, "O": no error.

3.3 Learning and Prediction for UMRSpell

During the pre-training, UMRSpell is driven by optimizing objectives of both detection and correction classifiers together:

$$L_d = \mathrm{CELoss}(\mathrm{Y_{tag}}, \tilde{\mathrm{Y}}_{\mathrm{tag}}) \quad (3)$$

$$L_c = \mathrm{CELoss}(\mathrm{Y_{text}}, \tilde{\mathrm{Y}}_{\mathrm{text}}) \quad (4)$$

where, Y is the input and $\tilde{Y}$ is the target. L_d and L_c are the losses of detection and correction parts, respectively. $\mathrm{CELoss}(\cdot)$ means cross entropy loss, the criterion[3] for it can be described as:

$$loss = -\sum_{c=1}^{C} y_c log \frac{exp(x_c)}{\sum_{i=1}^{C} exp(x_i)} \quad (5)$$

where x is the input, y is the target, and C is the number of classes. The overall objective L is defined as:

$$L = \lambda L_d + (1-\lambda) L_c \quad (6)$$

where, $\lambda \in [0, 1]$ is a coefficient to balance the detection and correction losses. Instead of manual coordination, λ is involved in gradient updating, and automatically optimizes itself with the whole network by taking use of AdamW (Loshchilov and Hutter, 2018):

$$\lambda = \begin{cases} 0.9 & 1 < \lambda \\ AdamW(\lambda) & if \quad 0 < \lambda < 1 \\ 0.1 & \lambda < 0 \end{cases} \quad (7)$$

Correspondingly, UMRSpell takes the same operations in its fine-tuning.

After learning, UMRSpell can be used as both detector and corrector in prediction process as shown in Figure 2. As a detector, UMRSpell outputs the detected tags for Sequence Re-tagging Rules unit and abandons the output corrected text. As a corrector, UMRSpell accepts an input text concatenated by both the original input and the re-tagged texts, and then makes the prediction for both detection and correction parts. With this approach, the corrector could handle the newly-tagged input that having different length with the output of detector.

3.4 Construction for ECMR-2023

To further investigate the effectiveness of UMRSpell in dealing with missing and redundant errors, a new dataset ECMR-2023 is constructed.

[3]torch.nn.CrossEntropyLoss of PyTorch is used: https://pytorch.org/docs/stable/index.html

Corpus Source Selection All 100,000 correct sentences from the publicly available competition dataset CTC-2021[4] are used as the original text. CTC-2021 is one of the most influential Chinese NLP competitions that is hosted by Chinese Association for Artificial Intelligence. It selects web texts written by native Chinese writers on the Internet as proofreading data. Further information can be seen in (Zhao et al., 2022).

Annotation Scheme Five character-level errors are tagged in texts ($\frac{num\ of\ err}{total\ err} \times 100\%$): *phonetic misspell* ($27.56\%$), *visual misspell* ($13.56\%$), *other misspell* ($39.77\%$), *missing error* ($6.37\%$), and *redundant error* (12.74%). The first three are spelling errors. According to (Liu et al., 2010), about 83% of errors are phonological and 48% are visual. However, we find that in real industrial process, missing and redundant errors are often deeply bound to the task, e.g., in customer service conversation scenarios. Therefore, we coordinate the proportion of five errors based on the probability of encountering them in practical applications.

Generalization Specially, we consider the following case: a synonym containing several wrong characters might replace the expected correct word because of a slip of the tongue during a conversation. Hence, we utilize word2vec of HanLP[5] to build *other misspell* samples following the steps: 1) Randomly select 1∼3 notional word from the current text; 2) Replace it with its closest neighbor in tensor space; 3) Proportionally choose one from the rest four error types to handle one token of the neighbor. With this approach, the whole dataset becomes more general.

Annotation Workflow We design an automatic process in Algorithm 1 to generate samples. Furthermore, a dictionary D containing Chinese similar phonetic and morphological characters integrated from (Ming, 2021) is used.

Quality Control Three experts manually check the generated dataset by randomly selecting 500, 100, 50 sentences for each errors in training, evaluation, and test sets, respectively. A sample will pass only if all inspectors agree. The inspection is repeated 5 times and completed only if the average pass rate reaches 98% or more. Detailed information of the final built ECMR-2023 is given in Table 1.

Algorithm 1: Paired detection and correction samples generation algorithm

Input: original correct text X_{true}, dictionary D, proportion of error types p, ratio $k\_p = 0.1 \in [0, 1]$, maximum number of selected tokens $k\_max = 3 \in [1, max\_seq\_len]$

Output: wrong text X_{wrong}, masked correct text X_{mask}

Tokenize X_{true} into X_{token};
Segment X_{true} into X_{word};
Calculate $k = min(len(X_{word}) \cdot k\_p, k\_max)$
Select k tokens to be the erroneous ones, randomly from X_{token} to form the list L;
foreach *token in* L **do**
    Select an action $flag$ from five kinds of errors following a proportion p;
    Update X_{token} via replacing/deleting/inserting operation with D according to $flag$;
end
Generate X_{wrong} and X_{mask} from X_{token}.

Number of #	Train	Dev.	Test
Phonetic misspell	31,212	3,093	1,068
Visual misspell	15,323	1,559	525
Other misspell	45,061	4,480	1,512
Missing error	7,292	646	246
Redundant error	14,464	1,432	456
Characters	1,538,694	151,730	19,130
Sentences	30,000	3,000	1,000

Table 1: Statistics of ECMR-2023, including the number of 5 kinds of errors, Chinese characters, and sentences.

4 Experiments

4.1 Settings

Datasets Dataset from (Wang et al., 2018) is used to pre-train UMRSpell, which has 271K sentences with 382K errors.[6] Both SIGHAN 13 (Wu et al., 2013), SIGHAN 14 (Yu et al., 2014), SIGHAN 15 (Tseng et al., 2015) benchmarks and ECMR-2023 are used to fine-tune and evaluate all participated models.

Evaluation Metrics To make more comprehensive evaluation, both of the sentence-level and character-level precision, recall, and F1-score are reported as the evaluation metrics as in (Cheng

[4]Open source: https://github.com/destwang/CTC2021
[5]Official website: https://hanlp.hankcs.com/docs/index.html
[6]Open source: https://github.com/wdimmy/Automatic-Corpus-Generation

Data	Methods	Sentence-level						Character-level					
		Detection			Correction			Detection			Correction		
		R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)
SIGHAN 13	BERT	71.7	77.6	74.5	47.4	51.3	49.3	82.0	67.9	74.3	76.1	63.1	69.0
	SpellGCN	$47.0_{74.4}$	$56.9_{80.1}$	$51.1_{77.2}$	$67.2_{72.7}$	$74.4_{78.3}$	$70.6_{75.4}$	$-_{88.9}$	$-_{82.6}$	$-_{85.7}$	$-_{88.4}$	$-_{98.4}$	$-_{93.1}$
	ChunkM	**75.7**	61.2	67.7	67.2	74.3	70.6	–	–	–	–	–	–
	FASPell	63.2	76.2	69.1	60.5	73.1	66.2	–	–	–	–	–	–
	PLOME	–	–	–	–	–	–	**89.3**	85.0	**87.1**	89.1	98.7	93.7
	Ours(p)	66.8	75.6	70.9	63.1	71.4	67.0	76.9	84.0	80.3	95.4	95.7	95.3
	Ours(p&f)	73.6	**83.0**	**78.0**	**71.0**	**80.0**	**75.2**	81.0	**89.2**	84.9	**96.4**	**96.7**	**96.4**
SIGHAN 14	BERT	**60.6**	72.0	65.8	41.8	49.7	45.4	62.5	75.0	68.2	59.3	71.1	64.7
	SpellGCN	$54.5_{69.5}$	$58.3_{65.1}$	$56.28_{67.2}$	$47.6_{67.2}$	$51.0_{63.1}$	$49.3_{65.3}$	$-_{78.6}$	$-_{83.6}$	$-_{81.0}$	$-_{76.4}$	$-_{97.2}$	$-_{85.5}$
	ChunkM	54.8	**78.7**	**64.6**	51.0	**77.4**	**61.5**	–	–	–	–	–	–
	FASPell	53.5	61.0	57.0	52.0	59.4	55.4	–	–	–	–	–	–
	PLOME	–	–	–	–	–	–	**79.8**	88.5	**83.9**	78.8	**98.8**	87.7
	Ours(p)	57.0	66.1	61.2	52.2	60.6	56.0	66.1	85.3	74.5	91.9	94.2	92.5
	Ours(p&f)	56.6	69.0	62.2	**57.2**	63.9	60.4	62.3	**88.6**	73.2	**92.6**	95.0	**93.3**
SIGHAN 15	BERT	68.4	84.1	75.4	54.2	66.0	59.7	65.5	83.4	73.4	62.8	79.9	70.3
	SMBERT	73.2	73.7	73.5	66.2	66.7	66.4	–	–	–	–	–	–
	SpellGCN	$64.0_{80.7}$	$71.0_{74.8}$	$67.3_{77.7}$	$54.18_{77.7}$	$60.1_{72.1}$	$57.0_{75.9}$	$-_{87.7}$	$-_{88.9}$	$-_{88.3}$	$-_{83.9}$	$-_{95.7}$	$-_{89.4}$
	ChunkM	62.0	**88.1**	72.8	57.6	**87.3**	69.4	–	–	–	–	–	–
	FASPell	60.0	67.6	63.5	59.1	66.6	62.6	–	–	–	–	–	–
	PLOME	**81.5**	77.4	**79.4**	**79.3**	75.3	**77.2**	**87.4**	**94.5**	**90.8**	84.3	97.2	90.3
	Ours(p)	67.7	76.1	71.7	60.2	67.8	63.8	71.2	90.6	79.8	91.2	**93.4**	91.5
	Ours(p&f)	72.2	77.2	75.0	64.8	69.3	67.0	75.1	92.4	83.0	**91.6**	92.8	**91.5**

Table 2: Performance of UMRSpell (Ours) and baseline models on SIGHAN series datasets, where R, P, F1 means recall, precision, F1-score, respectively. p and f behind UMRSpell means *pre-trained* and *fine-tuned*, respectively. Results of UMRSpell and BERT are from our implementation. Considering the fairness, BERT is first trained on (Wang et al., 2018) and then fine-tuned on the current dataset. Subscripts and main body of SpellGCN is from the original paper (Cheng et al., 2020) and its reproduced report (Bao et al., 2020), respectively.

et al., 2020; Liu et al., 2021). These metrics are provided for both detection and correction sub-tasks.

Hyper-parameter Settings We use BERT_{base} as the transformer encoder and keep the same settings with the original one (Devlin et al., 2019). We set the maximum sentence length to 128, batch size to 32 and the learning rate to 6e-5. These parameters are set based on experience because of the large cost of pre-training (on a single Tesla V100 (8×16G) server for nearly 12 hours). Better performance could be achieved if parameter tuning technique (e.g., grid search) is employed. Moreover, instead of training UMRSpell from scratch, we adopt the parameters of Chinese BERT released by Google[7] to initialize the Transformer blocks. For all experiments, we run our model five times and report the averages.

Baseline Models Correspond to **2 Related Work** in this paper, representative baselines from highly-relative types are compared with UMRSpell, including: Soft-masked BERT (SMBERT) (Zhang et al., 2020) of **Type-3**, SpellGCN (Cheng et al., 2020) of **Type-4**, Chunk-based Model (ChunkM) (Bao et al., 2020) of **Type-5**, FASPell (Hong et al., 2019) and PLOME (Liu et al., 2021) of **Type-6**. Moreover, BERT (Devlin et al., 2019) is also considered.

[7]Open source: https://github.com/google-research/bert

4.2 Results on SIGHAN series benchmarks

Table 2 illustrates the performance of UMRSpell and baseline models on SIGHAN 13∼15. From this table, we observe that: 1) UMRSpell ranks top 3 in most cases, especially achieving the best F1-scores in correction parts on all three benchmarks in character-level metric. Chunk-based Model (ChunkM) obtains outstanding performance on SIGHAN 14 in sentence-level metric, attributed to its global optimization to correct single- and multi-character typos (Bao et al., 2020). However, ChunkM depends on beam search algorithm to generate the correct text, which might be lower efficient than the other non-regression-based methods including UMRSpell. The other powerful competitors, PLOME and SpellGCN, design more task-specific complex networks to capture a priori knowledge or features of characters, even utilizing far more larger corpora (162.1 million sentences for PLOME (Liu et al., 2021)) in pre-training. Considering factors mentioned above, UMRSpell seems to keep its competitiveness. 2) UMRSpell with fine-tuning outperforms that without the process, where the difference of F1-scores between the two is less than 4%, implying the generalization ability of UMRSpell. 3) UMRSpell performs relatively

Data	Detector+Corrector	Sentence-level						Character-level					
		Detection			Correction			Detection			Correction		
		R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)	R (%)	P (%)	F1 (%)
SIGHAN 13	BERT+BERT	70.9	77.4	74.0	2.5	2.8	2.6	81.4	68.2	74.2	2.9	3.3	3.1
	Ours+BERT	69.5	77.1	73.1	2.4	2.5	2.5	80.2	68.5	73.9	5.3	6.9	6.0
	BERT+Ours	68.7	74.3	71.3	61.9	68.1	64.9	80.4	82.3	81.3	93.1	94.6	93.3
	Ours+Ours	**73.6**	**83.0**	**78.0**	**71.0**	**80.0**	**75.2**	**81.0**	**89.2**	**84.9**	**96.4**	**96.7**	**96.4**
SIGHAN 14	BERT+BERT	**62.4**	66.8	64.5	3.5	4.2	3.8	63.3	82.1	71.5	5.6	6.7	6.1
	Ours+BERT	55.4	67.2	60.7	3.9	4.3	4.1	62.1	83.5	71.2	5.3	6.9	6.0
	BERT+Ours	61.6	68.7	**64.9**	**54.7**	61.0	**57.8**	**68.2**	86.2	**76.3**	**88.6**	91.7	89.4
	Ours+Ours	56.6	**69.0**	62.2	52.3	**63.9**	57.6	62.3	**88.6**	73.2	92.6	**95.0**	**93.3**
SIGHAN 15	BERT+BERT	68.2	**78.5**	73.0	4.7	5.4	6.0	65.2	83.4	73.2	5.1	5.3	5.2
	Ours+BERT	68.6	73.7	71.1	4.3	5.0	4.6	66.2	82.5	73.5	4.7	5.0	4.8
	BERT+Ours	70.8	77.1	73.8	63.0	68.6	65.7	**77.2**	88.7	82.6	90.3	**93.0**	90.8
	Ours+Ours	**72.2**	77.2	**75.0**	**64.8**	**69.3**	**67.0**	75.1	**92.4**	**83.0**	**91.6**	92.8	**91.5**

Table 3: Performance of four kinds of detection-correction frameworks, where R, P, F1 means recall, precision, F1-score, respectively. All results are from our implementation. Considering the fairness, BERT is first fine-tuned on masked version of (Wang et al., 2018) and then fine-tuned on the current dataset. UMRSpell (Ours) is the pre-trained and fine-tuned version.

better in correction part than detection one, indicating that the designed Detection Transmission Self-attention is valid to pass more information of tokens in detection part to those in correction part.

4.3 Ablation Study

In this section, we validate whether UMRSpell works when it is used as a detector or a corrector during the prediction process, and analyze the reason behind that. To control variates, we replace UMRSpell to BERT from the whole prediction framework in Figure 2 in turn so as to get four different kinds of detector and corrector combinations listed in Table 3. From the table we find that: 1) UMRSpell+UMRSpell achieves the best result of all in most situations; 2) UMRSpell+UMRSpell outperforms BERT+UMRSpell, implying that UMRSpell works better at acquiring feature representations than BERT as a detector; 3) When BERT is used as a corrector, the results on correction task collapsed. Since BERT does not have specially-designed self-attention matrix that passes information from detection part to correction part as that in UMRSpell, it might not be good at learn the feature of masked errors. 4) Finally, UMRSpell is found highly adaptive to both detection and correction parts, which make it very flexible to use in real world situations.

4.4 Results on ECMR-2023

Table 4 shows the performance of the selected representative models on our proposed dataset. The overall trend of performance change is basically consistent with those on the SIGHAN 13∼15 benchmarks. It is found that there is still a lot of room for improvement of models on the dataset. New versions including more samples of more error types would be continuously updated for ECMR.

Model	Detection F1 (%)	Correction F1 (%)
BERT	56.5	33.2
PLOME	62.6	36.7
Ours (p&f)	68.2	54.6

Table 4: Performance of representative models on ECMR-2023 using character-level metric. Results of UMRSpell (Ours) and BERT are from our implementation. Considering the fairness, BERT is first fine-tuned on (Wang et al., 2018) and then fine-tuned on ECMR-2023. UMRSpell is the pre-trained and fine-tuned version.

5 Conclusion

In this paper, we propose UMRSpell to learn detection and correction parts together with a detection transmission self-attention matrix, and flexibly deal with Chinese missing, redundant, and spelling errors through re-tagging rules. We further construct a dataset ECMR-2023 containing five kinds of spelling errors to enrich the existing CSC task. Experiments on both SIGHAN benchmarks and ECMR-2023 demonstrate the significant effectiveness of UMRSpell.

Limitations

Currently, to deal with spelling, missing, redundant character errors in Chinese text, we jointly pre-train two sub-tasks based on a masked language model with task-specific attention mechanism and utilize re-tagging rules to reformulate the length of text during prediction. The proposed model might be less effective in more complex scenarios:

Word-Level case According to the structure of our model, it could theoretically handle errors that are not limited to character-level, such as redundant or missing words. However, the currently used self-attention matrix works between tokens instead of spans of tokens. A novel attention mask strategy might be considered. If the problem is solved, then our model would be able to handle both Chinese Spelling Correction (CSC) and some kinds of Grammatical Error Correction (GEC) tasks at the same time.

Task-specific Backbone case The backbone of the proposed model is BERT that is not task-specific, while some errors in SIGHAN happened in entities that might need priori knowledge to solve. For example, the correct sentence is "我要跟我的朋友去师大夜市" and the wrong sentence is "我要跟我的朋友去市大夜市", where the mispelled character belonging to an entity "师大" that is abbreviation of "师范大学" (means "Normal University"). To improve the performance of our model in more complicated applications, backbones that learn more task-specific knowledge should be considered.

Languages Mixture case In real world OCR or ASR applications, a Chinese character might be confused not only by another Chinese character, but also by an English character due to their similar pronunciation or shape. For example, the Chinese character "丁" is visually similar to the English capital letter "J", while the Chinese "喂" (means "Hi") is phoneticly similar to the English word "Way". Furthermore, a same character of simplified Chinese and traditional Chinese might be visually different.

High Efficiency case Industrial applications often require the prediction time in milliseconds-level under controlled usage of GPUs, which would bring troubles to large models. Distillation or truncating strategies might be a way to improve the proposed model.

Acknowledgements

This work was supported by the National Innovation 2030 Major S&T Project of China (No. 2020AAA0104200 & 2020AAA0104205). Linlin Wang was also supported by the National Natural Science Foundation of China (No. 62006077) and Shanghai Sailing Program (No. 20YF1411800).

References

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

Zuyi Bao, Chen Li, and Rui Wang. 2020. Chunk-based Chinese spelling check with global optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2031–2040, Online. Association for Computational Linguistics.

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

Chao-Huang Chang. 1995. A new approach for automatic chinese spelling correction. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, volume 95, pages 278–283. Citeseer.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of language modeling for Chinese spelling check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83, Nagoya, Japan. Asian Federation of Natural Language Processing.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. SpellGCN: Incorporating phonological and visual similarities into language models for Chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rahhal Errattahi, Asmaa El Hannani, and Hassan Ouahmane. 2018. Automatic speech recognition errors detection and correction: A review. *Procedia Computer Science*, 128:32–37.

Zifa Gan, Hongfei Xu, and Hongying Zan. 2021. Self-supervised curriculum learning for spelling error correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3487–3494, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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*.

Daniel Hládek, Ján Staš, and Matúš Pleva. 2020. Survey of automatic spelling correction. *Electronics*, 9(10):1670.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. FASPell: A fast, adaptable, simple, powerful Chinese spell checker based on DAE-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169, Hong Kong, China. Association for Computational Linguistics.

Li Huang, Junjie Li, Weiwei Jiang, Zhiyu Zhang, Minchuan Chen, Shaojun Wang, and Jing Xiao. 2021. PHMOSpell: Phonological and morphological knowledge guided Chinese spelling check. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5958–5967, Online. Association for Computational Linguistics.

Tuo Ji, Hang Yan, and Xipeng Qiu. 2021. SpellBERT: A lightweight pretrained model for Chinese spelling check. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3544–3551, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chong Li, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2021. Exploration and exploitation: Two ways to improve Chinese spelling correction models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 441–446, Online. Association for Computational Linguistics.

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified Chinese words. In *Coling 2010: Posters*, pages 739–747, Beijing, China. Coling 2010 Organizing Committee.

Shulin Liu, Shengkang Song, Tianchi Yue, Tao Yang, Huihui Cai, TingHao Yu, and Shengli Sun. 2022. CRASpell: A contextual typo robust approach to improve Chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3008–3018, Dublin, Ireland. Association for Computational Linguistics.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. PLOME: Pre-training with misspelled knowledge for Chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000, Online. Association for Computational Linguistics.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid Chinese spelling correction using language model and statistical machine translation with reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan. Asian Federation of Natural Language Processing.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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.

Xu Ming. 2021. Pycorrector: Text error correction tool. `https://github.com/shibing624/pycorrector`.

Xuan Ouyang, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE-M: Enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 27–38, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Arup Sarma and David D. Palmer. 2004. Context-based speech recognition error detection and correction. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 85–88, Boston, Massachusetts, USA. Association for Computational Linguistics.

DS Shih et al. 1992. A statistical method for locating typo in chinese sentences. *CCL Research Journal*, pages 19–26.

Yuen-Hsien Tseng, Lung-Hao Lee, Li-Ping Chang, and Hsin-Hsi Chen. 2015. Introduction to SIGHAN 2015 bake-off for Chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Baoxin Wang, Wanxiang Che, Dayong Wu, Shijin Wang, Guoping Hu, and Ting Liu. 2021. Dynamic connected networks for Chinese spelling check. In

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2437–2446, Online. Association for Computational Linguistics.

Baoxin Wang, Xingyi Duan, Dayong Wu, Wanxiang Che, Zhigang Chen, and Guoping Hu. 2022. CCTC: A cross-sentence Chinese text correction dataset for native speakers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3331–3341, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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, Brussels, Belgium. Association for Computational Linguistics.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for Chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785, Florence, Italy. Association for Computational Linguistics.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42, Nagoya, Japan. Asian Federation of Natural Language Processing.

Heng-Da Xu, Zhongli Li, Qingyu Zhou, Chao Li, Zizhen Wang, Yunbo Cao, Heyan Huang, and Xian-Ling Mao. 2021. Read, listen, and see: Leveraging multimodal information helps Chinese spell checking. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 716–728, Online. Association for Computational Linguistics.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of SIGHAN 2014 bake-off for Chinese spelling check. In *Proceedings of the Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132, Wuhan, China. Association for Computational Linguistics.

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang, Zhongjun He, Yu Sun, Hua Wu, and Haifeng Wang. 2021. Correcting Chinese spelling errors with phonetic pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2250–2261, Online. Association for Computational Linguistics.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 882–890, Online. Association for Computational Linguistics.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. MuCGEC: a multi-reference multi-source evaluation dataset for Chinese grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. Association for Computational Linguistics.

Honghong Zhao, Baoxin Wang, Dayong Wu, Wanxiang Che, Zhigang Chen, and Shijin Wang. 2022. Overview of ctc 2021: Chinese text correction for native speakers. *arXiv preprint arXiv:2208.05681*.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 49–56, Osaka, Japan. The COLING 2016 Organizing Committee.

Liying Zheng, Yue Deng, Weishun Song, Liang Xu, and Jing Xiao. 2021. An alignment-agnostic model for Chinese text error correction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 321–326, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chenxi Zhu, Ziqiang Ying, Boyu Zhang, and Feng Mao. 2022. MDCSpell: A multi-task detector-corrector framework for Chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1244–1253, Dublin, Ireland. Association for Computational Linguistics.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Please see Section "Limitations" after the text.

☒ A2. Did you discuss any potential risks of your work?
There is no potential risks mentioned in "Responsible NLP Research checklist guidelines - A2" in our work.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Please see "Abstract" and Section 1 " Introduction".

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☒ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
No response.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

Please see Section 4 "Experiments" - "4.1 Settings".

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Please see Section 4 "Experiments" - "4.1 Settings" - "Hyper-parameter Settings".

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☒ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Please see Section 4 "Experiments" - "4.1 Settings" - "Hyper-parameter Settings ": "These parameters are set based on experience because of the large cost of pre-training. Better performance could be achieved if parameter tuning technique (e.g., grid search) is employed."

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Please see Section 4 "Experiments" - "4.2 Results on SIGHAN series benchmarks" and "4.3 Ablation Study", bold fonts are used in the tables to highlight the best methods.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Please see footnote 3 7.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.