PTCSpell: Pre-trained Corrector Based on Character Shape and Pinyin for Chinese Spelling Correction

Xiao Wei[1]*, Jianbao Huang[1]*, Hang Yu[1]†, Qian Liu[2]
[1]Shanghai University
[2]Nanyang Technological University, Singapore
{xwei,845514379,yuhang}@shu.edu.cn, liu.qian@ntu.edu.sg

Abstract

Chinese spelling correction (CSC) is a challenging task with the goal of correcting each wrong character in Chinese texts. Incorrect characters in a Chinese text are mainly due to the similar shape and similar pronunciation of Chinese characters. Recently, the paradigm of pre-training and fine-tuning has achieved remarkable success in natural language processing. However, the pre-training objectives in existing methods are not tailored for the CSC task since they neglect the visual and phonetic properties of characters, resulting in suboptimal spelling correction. In this work, we propose to pre-train a new corrector named PTCSpell for the CSC task under the detector-corrector architecture. The corrector we propose has the following two improvements. First, we design two novel pre-training objectives to capture pronunciation and shape information in Chinese characters. Second, we propose a new strategy to tackle the issue that the detector's prediction results mislead the corrector by balancing the loss of wrong characters and correct characters. Experiments on three benchmarks (i.e., SIGHAN 2013, 2014, and 2015) show that our model achieves an average of 5.8% F1 improvements at the correction level over state-of-the-art methods, verifying its effectiveness.

1 Introduction

Chinese spelling correction (CSC) is a task which detects incorrect characters in Chinese text and corrects them. CSC is often used as post-processing to ensure the quality of a search engine query text (Gao et al., 2010; Duan et al., 2018) and academic papers (Pollock and Zamora, 1984). CSC plays an important role in correcting recognition errors due to similar pronunciation and character shape (Park et al., 2021; Nguyen et al., 2021), which is a common issue with automatic speech recognition (ASR) and optical character recognition (OCR) systems.

Type	Sentence	Correction
Phono-logical	今天教师(shi1)里面很热。 Trans.: It's very hot in the teacher today.	室(shi4) classroom
Visual	操场上有一于(yu2)个人。 Trans.: There are a at people in the gym.	千(qian1) thousand

Table 1: Examples of incorrect character recognition due to similar pronunciation and shape.

Compared with other languages, Chinese has distinct characteristics, such as its unique pronunciation system (usually represented as pinyin) and writing norms, which often lead to the two problems, namely the same pronunciation may correspond to multiple characters and different characters have similar shapes. According to Liu et al. (2010), around 83% and 48% of errors in Chinese texts can be attributed to phonological and visual similarity respectively. The first sentence in Table 1 is an example of a character with a similar pronunciation, where "室" is misspelled as "师", and the second sentence is an example of a character with a similar shape, where "千" is misspelled as "于".

Chinese spelling correction is challenging because different people have different writing habits, resulting in a variety of mistakes for each character. As such, it is difficult for previous rule-based methods to address these issues effectively. Most of the recent works based on large pre-trained language models (Zhang et al., 2020; Zheng et al., 2021) perform well in the CSC task. For example, Guo et al. (2021); Liu et al. (2021) used artificially constructed confusion sets to pre-train language models for the CSC task. However, pre-training objectives tailored for CSC have not yet been explored. The visual and phonetic properties of characters are not fully considered in the pre-training process. Moreover, most works are based on the architecture of the detector-corrector (Zhang et al., 2020; Li et al., 2021; Zhu et al., 2022). However,

*Xiao Wei and Jianbao Huang contributed equally.
†Hang Yu is the corresponding author.

Findings of the Association for Computational Linguistics: ACL 2023, pages 6330–6343
July 9-14, 2023

Sentence	Correction
我的录影(ying3)机在哪?	音(yin1)
Where is my video recorder?	tape
晚上的花园很安(an1)静。	宁(ning2)
The garden is very quiet at night.	peaceful

Table 2: Examples of correct characters being incorrectly corrected.

an inherent problem occurs if the detector predicts a correct character as being wrong, in which case the corrector may change the correct character to another one which is also a reasonable response. For example, in Table 2, the correct character "影" is replaced by "音", and the correct character "安" is replaced by "宁". It can be seen that the modified sentence changes the semantics of the original text.

To solve these problems, we propose a pre-trained corrector based on the visual and pronunciation features of characters for the CSC task. By doing so, the corrector can capture the phonetic similarity and visual similarity between characters, and such a pre-training strategy is more tailored to the CSC task. To address the problem of the detector's prediction results misleading the corrector, our basic idea is to strengthen the ability of the corrector to recognize correct characters to ease the errors caused by detector.

In this work, we propose a two-stage pre-trained corrector. In the first stage of pre-training, the visual and phonological features of characters are taken into account, and the pre-training strategies matching CSC are designed, respectively from similar vision to character and from similar pinyin to character. The pre-trained models are denoted as similar character shape BERT (SCSBERT) and similar pinyin BERT (SPBERT). In the second stage of pre-training, three different pre-trained models, SCSBERT, BERT, and SPBERT are fully fused. In addition, we propose a novel loss function to fine-tune the corrector. We calculate the loss of a small number of correct characters and the loss of all the wrong characters in the original text, so that it not only corrects the wrong characters, it also prevents the correct characters from being corrected by mistake.

To verify the effectiveness of our method, we conduct experiments with our model on the SIGHAN 2013, 2014 and 2015 test set using the official SIGHAN testing tool, which achieve an average improvement in F1 of 5.2% and 5.8% at the detection-level and correction-level compared to the latest works, MDCSpell (Zhu et al., 2022) and REALISE (Xu et al., 2021).

Our main contributions are summarized as follows: (i) We propose a pre-trained corrector which is tailored for the CSC task; (ii) We design two pre-training objectives based on vision and pronunciation to enable the corrector to capture the shape and pinyin similarity between characters; (iii) we propose a new strategy to solve the problem where the prediction results of the detector mislead the corrector by balancing the loss of incorrect characters and correct characters.

2 Related Work

CSC is the task of detecting and correcting wrong characters in Chinese sentences. The previous works were mainly based on the n-gram language model, rule-based method and confusion set for character error detection and correction (Yeh et al., 2013; Chang et al., 2015; Chu and Lin, 2015). After this, the CSC task is usually transformed into a sequence tagging task. Some machine learning and deep learning methods such as CRF and Bi-LSTM have been used to classify each character in a text (Chang et al., 2015; Wang et al., 2018). These methods detect characters and select a character with the highest probability of being correct.

Large pre-trained language models (PTMs) based on Transformer (Vaswani et al., 2017), such as BERT (Devlin et al., 2019), XLNET (Yang et al., 2019) and SpanBERT (Joshi et al., 2020) have been proposed and are being increasingly used for CSC tasks. In Soft-masked BERT (Zhang et al., 2020), BERT is used as a corrector to correct wrong positions predicted by the gate recurrent unit (GRU). Hong et al. (2019) set different masking schemes to fine-tune BERT, and selected the optimal candidate results as the error correction results. In addition to using BERT as a corrector, ELECTRA's discriminator (Clark et al., 2020) is also used to detect errors (Zheng et al., 2021). Although BERT can be used in CSC tasks and achieves good results, the similarity between the target character and original character is not easy to learn. Therefore, a confusion set (LEEa et al., 2019) is used to pre-train the BERT-like model to solve the above problem. Guo et al. (2021) used artificially constructed confusion sets to pre-train BERT, which makes BERT more capable of correcting phonologically and visually similar characters. Liu et al. (2021) set different proportions to select phonologically and visually

similar characters in the process of using a confusion set for training.

Chinese spelling errors are mainly caused by characters which are similar in shape and pronunciation. Recent works focus on how to make full use of visual and phonetic features to improve the correction of these two types of errors. To correct errors due to characters having a similar shape, the Chinese characters are encoded into images or split into strokes. To correct errors due to characters having a similar pronunciation, the original text is converted into pinyin or speech features (Wang et al., 2018; Xu et al., 2021; Liu et al., 2021).

Although prior research has achieved good results on the CSC task, the following two problems remain unresolved. First, the visual and phonetic features of similar Chinese characters cannot be directly fused into BERT for pre-training. Second, in relation to the detector-corrector architecture, the corrector cannot effectively alleviate the error caused by detector. To solve these problems, we propose a pre-trained language model based on visual features and pinyin features and a special loss function for the corrector.

3 Methodology

In this section, we briefly introduce the preliminaries of the our method, then we detail the proposed PTCSpell.

3.1 Preliminaries

Task Definition The CSC task is defined as follows. Given a piece of Chinese text $X = (x_1, x_2, x_3, \ldots, x_n)$ which may contain some errors, the target is to transform it to a corrected text $Y = (y_1, y_2, y_3, \ldots, y_n)$. This task is generally formed as a mapping of the original sequence X to the corrected sequence Y, i.e., $f(X) = Y$. The particularity lies in that there are usually only a small number of wrong characters and the wrong character $x_i \in X$ has some similarity to its correct character $y_i \in Y$.

Architecture The mainstream paradigm for the CSC task is based on the detector-corrector architecture (Zhang et al., 2020; Li et al., 2021). The *detector* identifies whether each character is correct or wrong, and the *corrector* generates corrections for the detected errors. Our method is designed based on this architecture. We briefly introduce the basic detector and corrector networks used in our method, and our motivation for designing our PTCSpell method.

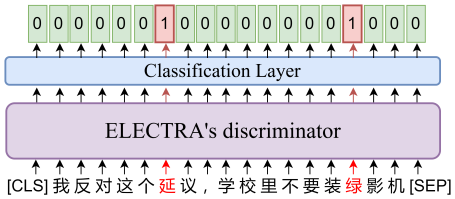

Figure 1: The architecture of the detection network.

Detector The detection network used in our model is based on the ELECTRA discriminator (Clark et al., 2020), as shown in Figure 1. Following Zheng et al. (2021); Li et al. (2021), error detection is defined as a character-level binary classification task. We employ the pre-trained Chinese ELECTRA* to initialize the parameters of the discriminator. The input sequence X is represented as character-level tokens, and then ELECTRA's discriminator encodes them to $\boldsymbol{H^d}$. The classification layer can be expressed as follows:

$$\boldsymbol{H_1} = GELU(\boldsymbol{W_1 H^d} + \boldsymbol{b_1}) \quad (1)$$

$$\boldsymbol{H_2} = LayerNorm(\boldsymbol{H_1}) \quad (2)$$

$$\boldsymbol{H_3} = \boldsymbol{W_2 H_2} + \boldsymbol{b_2} \quad (3)$$

where *GELU* is an activation function proposed by Hendrycks and Gimpel (2016), *LayerNorm* is the layer normalization proposed by Ba et al. (2016), $\boldsymbol{W_1} \in \boldsymbol{R}^{d \times d}$ and $\boldsymbol{W_2} \in \boldsymbol{R}^{1 \times d}$ (d is the size of the hidden states from ELECTRA's discriminator) are trainable parameters, b_1 and b_2 are bias vectors. $\boldsymbol{H_3} = \{\boldsymbol{h_1}, \boldsymbol{h_2}, \ldots, \boldsymbol{h_n}\}$ is the classification layer's output representation of each character. The probability that each character may be wrong can be defined as:

$$P^d(g_i = 1|X) = sigmoid(\boldsymbol{h_i}) \quad (4)$$

where *sigmoid* is the activation function, $P^d(g_i = 1|X)$ is a conditional probability indicating the probability that x_i is an error character.

Corrector The correction network is usually defined as a multi-class classification task at the token level, which is used to correct the characters at the location where the detection network output is "1", as shown in Figure 1. The correction network in some recent works is based on BERT with masked

*We use the released *chinese-electra-180g-base-discriminator* on Hugging Face https://huggingface.co/hfl/chinese-electra-180g-base-discriminator.

language model (MLM) task (Hong et al., 2019; Zhang et al., 2020). That is, each wrong character in the original text X detected by the detector is replaced with the [MASK] token, then the model predicts the most likely correct character to replace the [MASK] token.

However, it is important to observe the wrong characters in the original text for the corrector. For example, if people can read wrong characters in context, then it will be more conducive to correct them from a similar shape or similar pronunciation. As such, the disadvantage of the previous works is that the original text is lost when correcting, which may lead to deviation from the semantics of the original text.

Motivations The contribution of this work is mainly in relation to the correction network, which includes the following two aspects as follows. Firstly, we pre-train the correction network using similar character shape and similar pinyin. Secondly, we fine-tune the correction network under a special strategy which balances the loss of correct characters and wrong characters.

To solve the problem in the correction network caused by masked text, Zhu et al. (2022) use original text as input instead of masked text, which is also used in our architecture. In addition, we pre-train the correction network from similar character shape and similar pinyin. These pre-training objectives reduce the gap between pre-training and fine-tuning, and make the error correction process more targeted to two common types of spelling errors.

Another inherent problem caused by the architecture of the detector-corrector is that the detector may predict a correct character as the wrong character, which misleads the corrector. To alleviate the detector prediction error, we not only calculate the loss of wrong characters, but also correct characters during the fine-tuning of the corrector.

3.2 PTCSpell

We propose a pre-trained corrector based on character shape and pinyin for Chinese spelling correction, named PTCSpell. The architecture of the correction network is shown in Figure 2. It consists of three modules, i.e., Similar Character Shape BERT (SCSBERT), BERT and Similar Pinyin BERT (SPBERT), where SCSBERT and SPBERT are new designs. As suggested by the name of SCSBERT and SPBERT, the architecture of these two modules is exactly the same as BERT, both of which comprise 12 transformer blocks and 12 self-attention heads. The main difference between them is the pre-training objectives, i.e., SCSBERT and SPBERT are pre-trained by our designed pre-training objectives from scratch, while BERT is simply initialized by Chinese BERT with whole word masking[†] (Cui et al., 2021). More specifically, our proposed pre-training strategies are that SCSBERT is pre-trained by the visual features of characters and SPBERT is pre-trained by the phonetic features of characters (more details are given in Section 3.3).

The input of SCSBERT and BERT is the original text, while SPBERT is pinyin, which is converted from the original text using the pypinyin tool[‡]. Formally, given input X, SCSBERT, BERT and SPBERT encode it as $\boldsymbol{H^c}$, $\boldsymbol{H^b}$ and $\boldsymbol{H^p}$, respectively.

Then we consider how to fuse them to generate a unified representation for X. Since the character and pinyin at each position correspond to each other, it is felicitous to use the concatenation operator to fuse different features. In addition, the operator is conducive to preserving all information about the character and pinyin, thus it is expressed as follows:

$$\boldsymbol{H^{fused}} = Concat(\boldsymbol{H^c}, \boldsymbol{H^b}, \boldsymbol{H^p}) \quad (5)$$

where $\boldsymbol{H^c}$, $\boldsymbol{H^b}$, $\boldsymbol{H^p}$ are the last hidden state of SCSBERT, BERT and SPBERT, respectively, $\boldsymbol{H^c} \in \boldsymbol{R}^{b\times m\times d}$, $\boldsymbol{H^b} \in \boldsymbol{R}^{b\times m\times d}$, $\boldsymbol{H^p} \in \boldsymbol{R}^{b\times m\times d}$, $\boldsymbol{H^{fused}} \in \boldsymbol{R}^{b\times m\times 3d}$. b is the batch size, m is the maximum length of the text in the batch. d is the size of hidden states of BERT.

After this, we feed $\boldsymbol{H^{fused}}$ to the classification layer. The formula expression of the classification layer is the same as (1), (2) and (3), except for the dimension of $\boldsymbol{W}$, i.e., $\boldsymbol{W_1} \in \boldsymbol{R}^{3d\times 3d}$, $\boldsymbol{W_2} \in \boldsymbol{R}^{v\times 3d}$, where v is the size of the vocabulary. The final output of the classification layer is $\boldsymbol{h'_j}$, where j is the position of the character to be corrected.

$$P^c(y_j|X) = softmax(\boldsymbol{h'_j}) \quad (6)$$

where *softmax* is the activate function, and $P^c(y_j|X)$ indicates the probability that x_j is corrected to y_j.

[†]We use the released *chinese-bert-wwm-ext* on Hugging Face https://huggingface.co/hfl/chinese-bert-wwm-ext

[‡]We use the released *pypinyin* to convert Chinese text to pinyin https://github.com/mozillazg/python-pinyin.

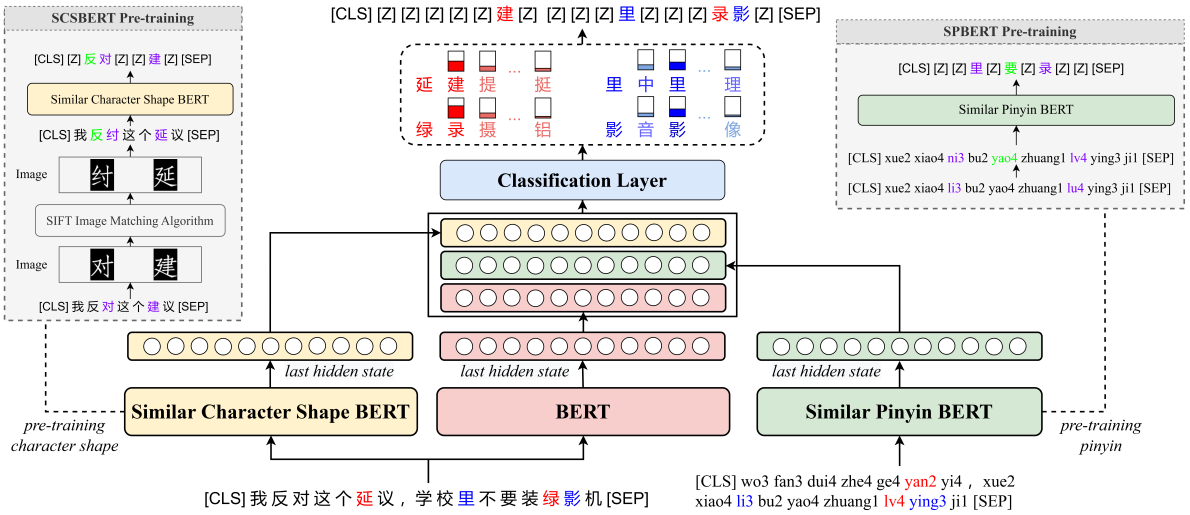

Figure 2: The architecture of the correction network. The correction network consists of three modules, all of which are pre-trained. SCSBERT is pre-trained in the way of vision to character (the left side), while SPBERT is pre-trained in the way of pinyin to character (the right side). The red and blue characters indicate that the detection network's prediction result of these are the wrong characters, while the red characters are true wrong characters and the blue characters are false wrong characters in the input text. The green and purple characters indicate two pre-training strategies (Section 3.3) in the first stage of pre-training.

3.3 Pre-training

There are two stages of pre-training in the training correction network: the first stage is pre-training SCSBERT and SPBERT from scratch, the second stage is pre-training the whole correction network. The purpose of the first stage of pre-training is to ensure SCSBERT and SPBERT have the ability to correct characters with similar shape and pronunciation, respectively. The second stage of pre-training can fully fuse the features extracted by SCSBERT, BERT and SPBERT.

SCSBERT To obtain the shape similarity of characters visually, we employ the SIFT (Lowe, 2004) image matching algorithm to calculate the visual similarity between two characters. Then, the characters with higher visual similarity are selected to replace the correct characters in the corpus to construct samples which have a similar character shape. As shown in Figure 2 (*SCSBERT Pre-training*), we use samples which have a similar character shape and the original text of the corpus as the pre-training dataset for SCSBERT.

SPBERT *SPBERT Pre-training* is illustrated in Figure 2. Firstly, we convert the prepared corpus text to pinyin, and then randomly replace some pinyin tokens with similar pronunciation ones. The sequence with the replaced pinyin is the input text X, and the original corpus data is the corrected text. The pre-training process of SPBERT converts a similar pinyin to a correct character.

Both the first stage of pre-training and the second stage of pre-training use the following two strategies: (i) randomly replace 10% of tokens of the text with similar characters or similar pinyin (the purple characters as shown in Figure 2); (ii) randomly select 4% of tokens of the text each time, and leave them unchanged (the green characters as shown in Figure 2).

3.4 Fine-tuning

Detection and correction tasks are defined as character classification. The difference is that the detection task is a binary classification task, while the correction task is a multi-classification task, where the number of classes is the size of the vocabulary. The loss function of the detection network is defined as:

$$L^d = -\sum_{i=1}^{n} \log P^d(g_i|X) \tag{7}$$

The correction network loss includes two parts, i.e., the loss of correct characters (the blue characters in Figure 2) and the loss of wrong characters (the red characters in Figure 2) in the original text. Given $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, we denote a target set T_1 consisting of correct characters in the original text. More specifically, we select characters from Y that are

equal to the characters in X.

$$T_1 = \{y_i | x_i = y_i, 1 \leq i \leq n, i \in N^*\} = \{t_1, t_2, \ldots, t_{m_1}\} \quad (8)$$

where m_1 is the size of T_1.

Characters of α proportion selected from T_1 are denoted as T_2. The set size is $m_2 = \alpha m_1$, where α is a pre-defined the hyper-parameters.

$$T_2 = \{t_i | t_i \in T_1, 1 \leq i \leq m_2, i \in N^*\} = \{t'_1, t'_2, \ldots, t'_{m_2}\} \quad (9)$$

To obtain the corrected characters corresponding to the wrong characters in the original text, we denote F as a set with size m_3:

$$F = \{y_i | x_i \neq y_i, 1 \leq i \leq n, i \in N^*\} = \{f_1, f_2, \ldots, f_{m_3}\} \quad (10)$$

Therefore, the loss of the correction network can be expressed as:

$$L^c = -\sum_{i=1}^{m_2} \log P^c(t'_i | X) - \sum_{i=1}^{m_3} \log P^c(f_i | X) \quad (11)$$

We train the detection network and correction network by minimizing L^d and L^c.

4 Experiments

4.1 Datasets and Metrics

To make a fair comparison with previous works, we use SIGHAN training data (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) and the generated pseudo data (Wang et al., 2018) as the datasets for pre-training and fine-tuning. We use the official SIGHAN2015 testing tool to evaluate the performance of our method in the three SIGHAN test sets. Following previous works, we use sentence-level precision, recall, and F1 as the metrics.

Our data preprocessing methods include converting traditional characters to simplified characters in the whole dataset using OpenCC tool[§] and removing a few mislabeled data in the SIGHAN2014 and SIGHAN2015 training sets. The SIGHAN dataset is repeated 5 times and joined with the Wang271K dataset as the final training set. The results are shown in Table 3.

Training Set	#Sent	Avg.Length	#Errors
SIGHAN2013	700	41.8	343
SIGHAN2014	3434	49.6	5140
SIGHAN2015	2337	31.3	3046
Wang217K	271329	42.6	391962
Total	303684	42.5	424607
Test Set	**#Sent**	**Avg.Length**	**#Errors**
SIGHAN2013	1000	74.3	1224
SIGHAN2014	1062	50.0	771
SIGHAN2015	1100	30.6	703
Total	3162	50.9	2698

Table 3: Statistics of Datasets. Column #Errors represents the number of character-level errors in this dataset. The total training set is SIGHAN repeated 5 times and joined with Wang271K.

4.2 Implementation Details

We fine-tune the detection network with 10 epochs, in which the learning rate is 1.5e-5 and the batch size is 8. The learning process of the correction network consists of two stages of pre-training and one stage of fine-tuning. We train the correction network with 10 epochs in the three stages, in which the learning rate is 2e-5 and the batch size is 8. The optimizer for the detection network and the correction network is AdamW (Loshchilov and Hutter, 2018). We use a warm-up (He et al., 2016) strategy to adjust the learning rate. Specifically, the learning rate increases linearly in the first quarter of iterations, but decreases linearly in the next three quarters of iterations. Following previous works (Xu et al., 2021), "的", "地" and "得" in SIGHAN2013 are not considered during evaluating because they are mixed in the dataset.

4.3 Baselines

We compare our model with the following methods:

- SpellGCN (Cheng et al., 2020): Graph convolutional networks are used to incorporate phonological and visual similarity knowledge into BERT.
- GAD (Guo et al., 2021): This method captures rich global context information to reduce the impact of local error context information.
- DCN (Wang et al., 2021): The candidate Chinese characters are generated by pinyin and then an attention-based network is used to model the dependencies between two adjacent characters.
- REALISE (Xu et al., 2021): This method selectively mixes the semantic, phonetic and graphic information of Chinese characters.

[§] https://github.com/BYVoid/OpenCC

Dataset	Method	Detection Level			Correction Level		
		Prec.	Rec.	F1.	Prec.	Rec.	F1.
SIGHAN2013	SpellGCN (Cheng et al., 2020)	80.1	74.4	77.2	78.3	72.7	75.4
	GAD (Guo et al., 2021)	85.7	79.5	82.5	84.9	78.7	81.6
	DCN (Wang et al., 2021)	86.8	79.6	83.0	84.7	77.7	81.0
	REALISE★(Xu et al., 2021)	88.6	**82.5**	85.4	87.2	**81.2**	84.1
	MDCSpell (Zhu et al., 2022)	89.1	78.3	83.4	87.5	76.8	81.8
	ELECTRA+BERT★ (baseline)	99.3	75.6	85.8	99.3	76.0	86.1
	PTCSpell (ours)★	**99.7**	80.6	**89.1**	**99.7**	79.2	**88.3**
SIGHAN2014	SpellGCN (Cheng et al., 2020)	65.1	69.5	67.2	63.1	67.2	65.3
	GAD (Guo et al., 2021)	66.6	**71.8**	69.1	65.0	**70.1**	67.5
	DCN (Wang et al., 2021)	67.4	70.4	68.9	65.8	68.7	67.2
	REALISE(Xu et al., 2021)	67.8	71.5	69.6	66.3	70.0	68.1
	MDCSpell (Zhu et al., 2022)	70.2	68.8	69.5	69.0	67.7	68.3
	ELECTRA+BERT (baseline)	65.4	68.1	66.7	64.0	64.0	64.0
	PTCSpell (ours)	**84.1**	71.2	**77.1**	**83.8**	69.4	**75.9**
SIGHAN2015	SpellGCN (Cheng et al., 2020)	74.8	80.7	77.7	72.1	77.7	75.9
	GAD (Guo et al., 2021)	75.6	80.4	77.9	73.2	77.8	75.4
	DCN (Wang et al., 2021)	77.1	80.9	79.0	74.5	78.2	76.3
	REALISE(Xu et al., 2021)	77.3	81.3	79.3	75.9	**79.9**	77.8
	MDCSpell (Zhu et al., 2022)	80.8	80.6	80.7	78.4	78.2	78.3
	ELECTRA+BERT (baseline)	75.5	**83.0**	79.1	74.6	79.0	76.7
	PTCSpell (ours)	**89.6**	81.2	**85.2**	**89.4**	79.0	**83.8**

Table 4: The performance of our model and the baselines. The symbol ★ means that we have removed the "的", "地" and "得" characters when evaluating SIGHAN2013.

- MDCSpell (Zhu et al., 2022): This method designs a multi-task framework, where BERT is used as a corrector to capture the visual and phonological features of the characters and integrated the hidden states of the detector to reduce the impact of error.

4.4 Results

As shown in Table 4, we observe that our PTCSpell achieves significant performance gain over the other baselines. It can be seen that the performance of our PTCSpell greatly exceeds that of ELECTRA+BERT. Specifically, at the correction level, our PTCSpell outperforms it in terms of F1 by 2.2%, 11.9% and 7.1% on the three SIGHAN test sets, respectively.

Then, we compare our PTCSpell with the most competitive works, such as MDCSpell and REALISE. At the correction level, our PTCSpell outperforms both of these in terms of F1 by 4.2%, 7.6% and 5.5%, respectively, and in terms of precision by 12.2%, 14.8% and 11% on the three SIGHAN test sets, respectively.

The main reasons are two-fold. First, PTCSpell learns the similarity between characters, thus the corrector selects characters which are similar to the original text in the correction process. Second, the novel loss function proposed on the corrector alleviates the problem of the detector predicting correct characters as errors. Although PTCSpell has made great improvements in precision and F1, recall is not improved compared with the baselines. A possible reason is that the detector fails to detect all errors, leading to a bottleneck in the corrector.

4.5 Ablation Study

We explore the influence of parameter α in the loss function on model performance and the contribution of the SCSBERT and SPBERT modules to the PTCSpell model. We evaluate our model at the sentence-level on the 2013, 2014 and 2015 SIGHAN test sets. The average F1 performance of the three test sets is shown in Table 5 and Table 6 (the detailed results are provided in Appendix A.1).

Table 5 shows the influence of different parameter α in the loss function on the performance of the PTCSpell model. We test the performance of our model by varying α in range of [0, 0.1] and [0.92, 1] with a step of 0.02. We find that a smaller α is better than a larger α. When α is 0, the model loses the ability to retain the original text semantics, so the model does not perform well. When α is 1, this means that all characters participate in the calculation of loss. It is observed that when α

α	Det. F1	Cor. F1
0	78.0	76.4
0.04	83.8	82.7
0.02-0.1	83.3	82.1
0.92-1	82.9	81.9
1	83.1	81.9

Table 5: Impact of different α in the loss function. (0.02-0.1) denotes the average F1 score of our model when alpha goes from 0.02 to 0.1 and the step size is 0.02. (0.92-1) denotes the average F1 score of our model when alpha goes from 0.92 to 1 and the step size is 0.02.

Model	α	Det. F1	Cor. F1
PTCSpell	0.04	83.8	82.7
PTCSpell(-PT)	0.04	83.3	82.2
SCSBERT+BERT(-PT)	0.04	83.1	82.0
BERT+SPBERT(-PT)	0.04	82.6	81.3
BERT(-PT)	0.04	82.3	81.2

Table 6: Results of the ablation experiment on PTCSpell. (-PT) denotes that the second stage of pre-training is removed.

is 0.04, the model achieves the best performance. Interestingly, according to our statistics, we find that wrong characters accounted for 3.3% of all characters in the training set. We infer that when α is set at 0.04, the number of correct characters and wrong characters is balanced, which is helpful to train the model.

Table 6 analyses the second stage of pre-training, SCSBERT and SPBERT for our PTCSpell. To verify the effectiveness of the pre-training in the second stage, we conduct experiments PTCSpell and PTCSpell(-PT). We find that the second stage of pre-training greatly improves the performance of the model. To verify the effectiveness of SCSBERT and SPBERT, we conduct experiments PTCSpell (-PT), SCSBERT + BERT (-PT), BERT + SPBERT (-PT) and BERT (-PT), where SCSBERT and SPBERT are pre-trained by character shape and pinyin, respectively. We find that SCSBERT and SPBERT both improve the performance of the model. Through these experiments, The effectiveness of SCSBERT, SPBERT and the second stage of pre-training are verified.

4.6 Case Study

Table 7 provides details on two cases of Chinese spelling correction. Given an input sentence, we compare the error correction effect of the baseline (ELECTRA + BERT) and PTCSpell. The detector (ELECTRA) detects character errors, where

Input	乍么才能让孩子对绘画蝉声兴趣呢?
Detector	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0
Baseline	什么才能让孩子对绘画蚊身兴趣呢?
PTCSpell	怎么才能让孩子对绘画产生兴趣呢?
Trans.	How to get children interested in painting?
Input	我们学校购买了十台录影机。
Detector	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0
Baseline	我们学校购买了十台录音机。
PTCSpell	我们学校购买了十台录影机。
Trans.	Our school has purchased ten video recorders.

Table 7: Two cases of Chinese spelling correction. The green characters indicate the true wrong characters and the detector predicts it to be "1". The blue characters indicate the true correct characters but the detector predicts it to be "1". The red character indicates the corrected character.

"0" means the current character is correct and "1" means the current character is wrong. If the output of the detector in the current position is "1", the character is corrected by the corrector. In the first case, "乍" is correctly predicted to be the wrong character, but the baseline corrects it to "什" and forms a phrase with the following "么", which fails to capture the character shape of "乍". Our PTCSpell corrects "乍" to "怎", because it captures the character shape successfully. For the other two wrong characters "蝉声", it is difficult for baseline to correct consecutive wrong characters, causing "蝉" to be corrected to "蚊" due to its similar shape. In contrast, our PTCSpell corrects them to "生产" based on similar pronunciations and is not affected by consecutive errors. The first case shows that PTCSpell achieves good performance in both similar shape and pronunciation errors. In the second case, "影" is a correct character but the detector predicts it to be a wrong character. We can see that the baseline does not handle this well, so it is replaced with "音", while PTCSpell leaves this character unchanged. The second case shows that PTCSpell is not misled by the detector because it learns whether the current character is really wrong and tries its best to maintain the semantics of the original sentence during error correction.

5 Conclusion

We propose a pre-trained corrector, which is a part of the detector-corrector architecture, for modeling similar shape and pronunciation errors in the CSC task. Our main contribution is to enhance the ability of the corrector. The architecture of PTCSpell is based on BERT, but is pre-trained from scratch based on similar character shapes and pinyin re-

spectively. In addition, we propose a special loss function that enhances the ability of the corrector to retain the correct characters in the original text. Finally, the experimental results show that our model is effective. In the future, we will design a better detector to further improve the recall score for the whole detector-corrector model.

Limitations

Our model achieves outstanding performance in relation to Chinese spelling correction. However, it has several potential limitations: (i) Errors of missing and redundant characters cannot be corrected by our model. The PTCSpell model only focuses on spelling errors, and requires that the input text has no grammatical or semantic errors. (ii) The error-correcting language is targeted at Chinese. The pre-trained model based on similar pinyin cannot adapt to other languages, while pre-trained model based on similar character shape can adapt to other languages well, because the pinyin input method is unique to Chinese, but character error due to a similar shape is a common problem in many languages. Nevertheless, we put forward the idea of matching the pre-trained model with error correction tasks, which is suitable for all languages.

Acknowledgments

This rescarch was supported by the Shanghai Science and Technology Young Talents Sailing Program (22YF1413600). We thank Maoxin Shen for helpful discussions, and the anonymous reviewers for their insightful comments.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Tao-Hsing Chang, Hsueh-Chih Chen, and Cheng-Han Yang. 2015. Introduction to a proofreading tool for chinese spelling check task of sighan-8. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 50–55.

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.

Wei-Cheng Chu and Chuan-Jie Lin. 2015. NTOU Chinese spelling check system in sighan-8 bake-off. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 137–143, Beijing, China. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514.

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.

Jianyong Duan, Tianxiao Ji, and Hao Wang. 2018. Error correction for search engine by mining bad case. *IEICE Transactions on Information and Systems*, E101.D(7):1938–1945.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 358–366, Beijing, China. Coling 2010 Organizing Committee.

Zhao Guo, Yuan Ni, Keqiang Wang, Wei Zhu, and Guotong Xie. 2021. Global attention decoder for Chinese spelling error correction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1419–1428, Online. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

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.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Lung-Hao LEEa, Wun-Syuan WUb, Jian-Hong LIa, Yu-Chi LINc, and Yuen-Hsien TSENG. 2019. Building a confused character set for chinese spell checking. In *27th International Conference on Computers in Education, ICCE 2019*, pages 703–705. Asia-Pacific Society for Computers in Education.

Jing Li, Gaosheng Wu, Dafei Yin, Haozhao Wang, and Yonggang Wang. 2021. Dcspell: A detector-corrector framework for chinese spelling error correction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1870–1874.

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, 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.

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

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. 2021. Survey of post-ocr processing approaches. *ACM Comput. Surv.*, 54(6).

Seongmin Park, Dongchan Shin, Sangyoun Paik, Subong Choi, Alena Kazakova, and Jihwa Lee. 2021. Improving distinction between asr errors and speech disfluencies with feature space interpolation. *arXiv preprint arXiv:2108.01812*.

Joseph J. Pollock and Antonio Zamora. 1984. Automatic spelling correction in scientific and scholarly text. *Commun. ACM*, 27(4):358368.

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.

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.

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Jui-Feng Yeh, Sheng-Feng Li, Mei-Rong Wu, Wen-Yi Chen, and Mao-Chuan Su. 2013. Chinese word spelling correction based on n-gram ranked inverted index list. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 43–48, Nagoya, Japan. Asian Federation of Natural Language Processing.

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.

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.

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.

A Appendix

A.1 Ablation

We show detailed results on the effect of different alpha on model performance. After two stages of pre-training, we set different alpha for the model training, and record the performance of the model on three SIGHAN test sets. The detailed results are shown in Table 8.

At the same time, we explore the contribution of SCSBERT and SPBERT to the whole model. In the following experiment, SCSBERT and SPBERT are pre-trained respectively, but the second stage pre-training is not carried out. Then they are fine-tuned, where the alpha in the loss function is set to 0.04. The detailed results are shown in Table 9.

Dataset	α	Detection Level			Correction Level		
		Prec.	Rec.	F1.	Prec.	Rec.	F1.
SIGHAN2013	0.00	99.34	78.59	87.75	99.32	76.40	86.37
	0.02	99.74	79.94	88.75	99.74	78.69	87.97
	0.04	99.74	80.56	89.13	99.74	79.21	88.30
	0.06	99.61	79.31	88.31	99.60	78.27	87.66
	0.08	99.61	79.73	88.57	99.61	78.79	87.99
	0.10	99.48	79.94	88.65	99.47	78.69	87.87
	0.92	99.61	79.21	88.25	99.60	78.27	87.66
	0.94	99.61	79.21	88.25	99.60	78.38	87.73
	0.96	99.61	79.21	88.25	99.60	78.48	87.79
	0.98	99.61	78.69	87.92	99.60	77.55	87.20
	1.00	99.61	79.83	88.63	99.61	78.90	88.05
SIGHAN2014	0.00	65.56	68.08	66.79	64.57	65.19	64.88
	0.02	83.18	71.35	76.81	82.84	69.62	75.65
	0.04	84.09	71.15	77.08	83.76	69.42	75.92
	0.06	85.51	69.23	76.51	85.19	67.50	75.32
	0.08	85.68	69.04	76.46	85.44	67.69	75.54
	0.10	84.16	68.46	75.50	83.78	66.54	74.17
	0.92	87.01	68.27	76.51	86.85	67.31	75.84
	0.94	85.68	66.73	75.03	85.43	65.38	74.07
	0.96	86.91	67.69	76.11	86.72	66.54	75.30
	0.98	85.17	68.46	75.91	84.91	67.12	74.97
	1.00	86.31	67.88	76.00	85.93	65.77	74.51
SIGHAN2015	0.00	76.40	83.03	79.58	75.74	80.07	77.85
	0.02	88.76	81.55	85.00	88.36	78.41	83.09
	0.04	89.61	81.18	85.19	89.35	78.97	83.84
	0.06	90.23	80.07	84.85	89.91	77.31	83.13
	0.08	89.02	80.81	84.72	88.70	78.23	83.14
	0.10	89.67	80.07	84.60	89.47	78.41	83.58
	0.92	89.94	79.15	84.20	89.61	76.38	82.47
	0.94	90.97	79.89	85.07	90.73	77.68	83.70
	0.96	90.27	78.78	84.14	89.98	76.20	82.52
	0.98	89.96	79.34	84.31	89.70	77.12	82.94
	1.00	90.72	79.34	84.65	90.43	76.75	83.03

Table 8: Impact of different α in the loss function.

Dataset	Model	Detection Level			Correction Level		
		Prec.	Rec.	F1.	Prec.	Rec.	F1.
SIGHAN2013	PTCSpell(-PT)	99.48	79.73	88.52	99.47	78.59	87.80
	SCSBERT+BERT(-PT)	99.74	80.46	89.07	99.74	79.21	88.30
	BERT+SPBERT(-PT)	99.61	79.94	88.70	99.61	78.69	87.92
	BERT(-PT)	99.61	79.42	88.37	99.60	78.27	87.66
SIGHAN2014	PTCSpell(-PT)	85.78	68.46	76.15	85.47	66.73	74.95
	SCSBERT+BERT(-PT)	85.44	68.85	76.25	85.12	67.12	75.05
	BERT+SPBERT(-PT)	82.83	68.65	75.08	82.34	66.35	73.48
	BERT(-PT)	82.24	67.69	74.26	81.86	65.96	73.06
SIGHAN2015	PTCSpell(-PT)	89.43	81.18	85.11	89.19	79.15	83.87
	SCSBERT+BERT(-PT)	88.57	80.07	84.11	88.26	77.68	82.63
	BERT+SPBERT(-PT)	88.03	80.07	83.86	87.76	78.04	82.62
	BERT(-PT)	89.09	79.89	84.24	88.79	77.49	82.76

Table 9: Results of ablation experiment on PTCSpell. (-PT) denotes that the second stage of pre-training is removed.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☑ A2. Did you discuss any potential risks of your work?
4.5 Ablation Study; Limitations

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract

☒ 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?
Footnote 2 and 3 on the second page.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Footnote 2 and 3 on the second page.

☑ 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)?
4 Experiments

☒ 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?
We use datasets published online.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Footnote 2 and 3 on the second page.

☑ 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.
4.1 Datasets and Metrics

C ☑ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4.2 Implementation Details

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?
4.2 Implementation Details

☑ 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?
4.5 Ablation Study

☑ 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.)?
4.2 Implementation Details

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.