MDCSpell: A Multi-task Detector-Corrector Framework for Chinese Spelling Correction

Chenxi Zhu, Ziqiang Ying, Boyu Zhang, Feng Mao
Alibaba Group
{mike.zcx, zhangboyu.zby, maofeng.mf}@alibaba-inc.com
yingzq0116@163.com

Abstract

Chinese Spelling Correction (CSC) is a task to detect and correct misspelled characters in Chinese texts. CSC is challenging since many Chinese characters are visually or phonologically similar but with quite different semantic meanings. Many recent works use BERT-based language models to directly correct each character of the input sentence. However, these methods can be sub-optimal since they correct every character of the sentence only by the context which is easily misled by the misspelled characters. Some other works propose to use an error detector to guide the correction by masking the detected errors. Nevertheless, these methods dampen the visual or phonological features from the misspelled characters which could be critical for correction. In this work, we propose a novel general detector-corrector multi-task framework where the corrector uses BERT to capture the visual and phonological features from each character in the raw sentence and uses a late fusion strategy to fuse the hidden states of the corrector with that of the detector to minimize the misleading impact from the misspelled characters. Comprehensive experiments on benchmarks demonstrate that our proposed method can significantly outperform the state-of-the-art methods in the CSC task.

1 Introduction

Chinese Spelling Correction (CSC) is a fundamental task that aims to automatically detect and correct spelling errors in Chinese texts. These spelling errors are typically caused by human writing, automatic speech recognition (ASR) or optical character recognition (OCR) systems (Afli et al., 2016; Wang et al., 2018). CSC is essential since it is crucial for many downstream tasks like search engine (Martins and Silva, 2004; Gao et al., 2010) and essay scoring (Burstein and Chodorow, 1999).

Despite its recent development, CSC remains a challenging task since many Chinese characters are visually or phonologically similar, but with great different semantic meanings. According to (Liu et al., 2010), around 83% and 48% of errors belong to phonological and visual similarity respectively. Moreover, the Chinese language usually consists of many characters without word deliminators, which makes the CSC system must recognize spelling errors based on contextual information, rather than just relying on individual words or characters.

Input	我这一次写信给你是想跟你安排一下关(guān)以(yǐ)我们要见面的。
baseline	我这一次写信给你是想跟你安排一下所(suǒ)以(yǐ)我们要见面的事。
Ground Truth	我这一次写信给你是想跟你安排一下关(guān)于(yú)我们要见面的事。
Translation	I am writing to you this time to make arrangements with you about our meeting.
Input	为了减少急遍(biàn)的生孩子率，需要呼吁适当的生育政策。
baseline	为了减少急速(sù)的生孩子率，需要呼吁适当的生育政策。
Ground Truth	为了减少急变(biàn)的生孩子率，需要呼吁适当的生育政策。
Translation	In order to reduce the rapidly changing rate of childbirth, it is necessary to call for an appropriate childbirth policy.

Table 1: Examples of CSC results, the incorrect and correct characters marked in red and blue respectively.

Many efforts have been put in the CSC task. Early methods are mainly based on the traditional language models (Liu et al., 2010, 2013; Yu and Li, 2014) or sequence-to-sequence models (Wang et al., 2019). Recently, with the emerge of pre-trained BERT model (Devlin et al., 2019), many methods have been proposed and made great progress in CSC. Most of these works like (Cheng et al., 2020; Guo et al., 2021; Wang et al., 2021) use BERT-based language models and confusion set to directly correct each character of the input sentence. However, these methods indistinguishably correct every character of the sentence via the contextual information which is easily misled by the misspelled characters. As shown in the upper case in Table 1, the context is affected by the error

Findings of the Association for Computational Linguistics: ACL 2022, pages 1244 - 1253

character “以” which makes the correction model mistakenly change the original correct character “关” to “所”. To address the above issues, some other works like (Hong et al., 2019; Zhang et al., 2020; Li et al., 2021) propose to use an error detector to detect the positions of errors which are used as prior knowledge for correction via masking. Nevertheless, these methods in turn would dampen the visual or phonological features from the misspelled characters which could be critical for correction. As illustrated in the lower case in Table 1, despite the model correctly finding the error position, it failed to change the error to correct character “变” since it misses the phonological feature from the misspelled character “遍”. Thus, how to exploit the visual and phonological features of the misspelled characters while expelling their misleading impact on the context still remains to be an open question in the CSC task.

To address the above issues, we propose a novel general multi-task detector-corrector CSC framework (MDCSpell) which can both employ the visual and phonological features of the misspelled characters while eliminating their misleading impact on the context. Specifically, the correction and detection tasks are executed simultaneously where the corrector uses BERT to capture the visual and phonological features of all characters directly from the raw sentence and the detector uses a light-weight transformer to detect the positions of misspelled characters. A late-fusion strategy is employed to fuse the hidden states of the corrector with that of the detector and enable the elimination of the misleading impact from the misspelled characters with an end-to-end joint training. This framework is simple to implement and any BERT-based CSC model can be easily adapted in this framework. Experimental results on three open benchmarks demonstrate that MDCSpell can significantly outperform the competitors.

In summary, our contributions are concluded as follows:

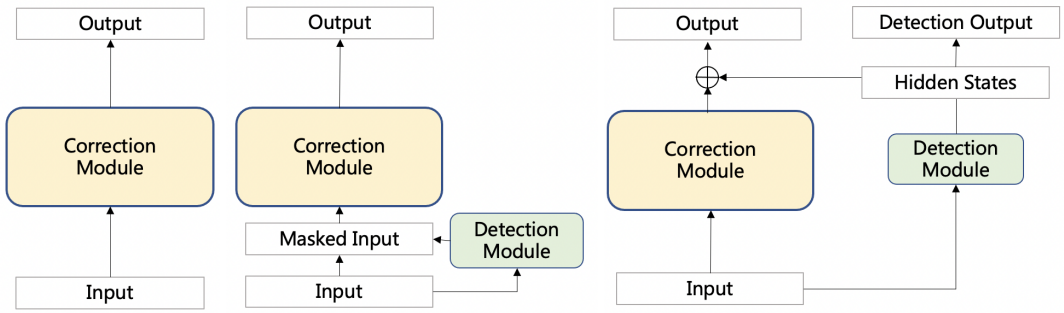
- We propose a novel general multi-task detector-corrector CSC framework MDCSpell which can both make use of the visual and phonological features of the misspelled characters which are critical for correction while minimizing their misleading impact on the context. The proposed framework is simple to implement and any BERT-based CSC models can be easily adapted in this framework.
- We investigate the performance of MDCSpell both quantitatively and qualitatively. The experimental results show the superiority of our method on three open benchmarks.

2 Related Work

Chinese spelling correction (CSC) is an important and challenging task. It mainly needs to detect the wrong characters based on the judgment of the semantics and correct these wrong characters with a full understanding of the context. Most of the early work used unsupervised language models and rules for detection and correction, and used the perplexity of language model for determination (Yeh et al., 2013; Yu and Li, 2014; Xie et al., 2015; Tseng et al., 2015). Recently a lot of works tend to transform CSC into a sequence tagging task, modeling each character in the sentence to determine the position of error and correct it into the right character(Wang et al., 2019; Ji et al., 2017; Chollampatt et al., 2016; Ge et al., 2018).

With the development of large-scale pretraining in NLP, an increasing number of works follow the way of solving sequence labeling problems, i.e., using the BERT-like model to directly map every character in the sentence to the correct ones. (Cheng et al., 2020) proposed a model named SpellGCN which incorporates phonological and visual similarity knowledge into BERT via a specialized graph convolutional network. (Huang et al., 2021) utilized phonological and morphological knowledge to model the similarities of the characters for correction. (Guo et al., 2021) proposed a global attention decoder that learns the global relationship of the potential correct input characters and the candidates of potential error characters. (Wang et al., 2021) proposed a dynamic connected network to model the dependencies between two adjacent Chinese characters to improve the corrector.

Some other works use an error detector as the preliminary for correction which turns the CSC into a two-stage pipeline. (Hong et al., 2019) proposed the FASpell model to predict candidate characters based on the BERT model and exploit the phonological and visual similarity information to select candidate characters. (Zhang et al., 2020) use a two-stage detection and correction method named Soft-Masked BERT, which masked the detected error characters with error probability and then turn the masked input into the BERT model for error correction. (Li et al., 2021) proposed a two-stage



(a) Direct correction (b) Correction with detected masks (c) The proposed scheme

Figure 1: A comparison between three schemes of CSC models. (a) Direct correction scheme directly correct each character of the input sentence without any error positions information. This scheme is vulnerable to the misleading impact on the context from the misspelled characters since the correction mainly relies on the context. (b) This scheme masks the input with detected error positions and predict the correct characters in the masked positions. This scheme can minimize the misleading impact from the misspelled characters, but dampen the useful visual and phonological features from them. (c) Our proposed scheme directly use the raw sentence as the input to the correction module to keep the visual and phonological features of the misspelled characters, and enabling the minimization of the misleading impact from them via the late fusion of hidden states from correction module and detection module.

cloze-style detector-corrector framework for correction.

Although the above methods have achieved good results on CSC tasks, they either suffer from the misleading effect on the context of the misspelled characters or miss the critical visual and phonological features of the spelling errors. In order to solve the above problems, we propose a multi-task learning architecture via late fusion to effectively use detection information for correction decision-making and improve the precision of the model.

3 Methodology

3.1 Problem Formulation

The Chinese Spelling Correction(CSC) task can be formalized as the following task. Given a text sequence of n Chinese characters $X = (x_1, x_1, \ldots, x_n)$, the goal is to output $Y = (y_1, y_2, \ldots, y_n)$, where X represents the original text containing some error characters, and Y represents the correct text after correction. The X and Y have the same length. Therefore, CSC task can be regarded as a sequence tagging task. Usually, no or only a small fraction of misspelled characters is in a sentence and all or most of the characters should be copied.

3.2 Motivations

Our motivation can be shown in the Figure 1. Most of the existing state-of-the-art CSC methods treat the correction as a sequence tagging task like Figure 1(a), that is to use the correction module to classify which character the corresponding token should be converted to. The disadvantage of this type of method is that they lack the awareness of the position of the misspelled characters and correct each characters merely by the context, which is easily misled by the misspelled characters.

In order to solve the problem, some methods (Hong et al., 2019; Zhang et al., 2020), as shown in Figure 1(b), add a detection module before the correction module to mask the positions where errors may occur and predict the correct characters in the masked positions. Although this scheme weakens the misleading impact of the spelling errors to a certain extent, it also leads to a new problem: the correction performance can still be sub-optimal since the phonological and visual information of the misspelled characters, which could be highly similar to the correct characters and helpful for correction, are dampened by the mask.

Therefore, the above issues inspire us to find a new scheme of utilizing the error detection information. Specifically, as shown in Figure 1(c), the raw sentence is directly used as the input of the correction module to keep the visual and phonological features of the misspelled characters, while the hidden states of the correction module are lately fused with those of the detection module. The misleading impact from the misspelled characters is minimized via the end-to-end joint training. In the following sections, we will illustrate how to implement the multi-task framework based on this scheme.

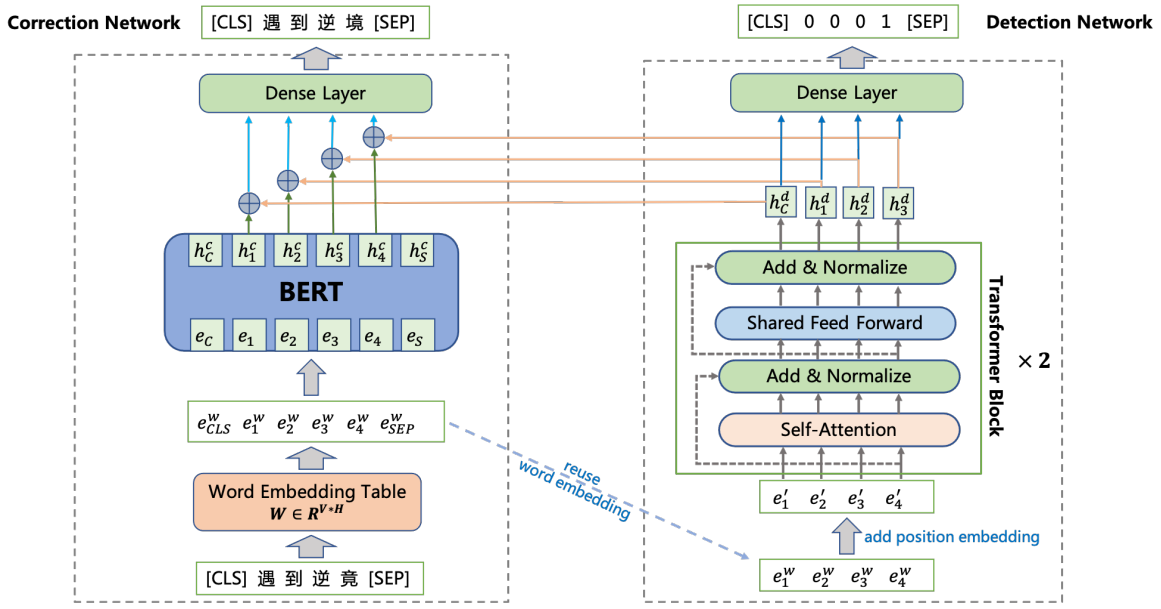


Figure 2: The overall structure of the MDCSpell. The MDCSpell uses a transformer structure as the detection network and a BERT structure as the correction network. These two networks share the same word embedding as input. At the end of the correction network, the hidden states from both the correction and detection networks are fused as input to the classification dense layer which generates the correction results. These two tasks can be trained simultaneously in an end-to-end manner.

3.3 Structure of MDCSpell

We implement the proposed correction scheme as the MDCSpell, which is depicted in Figure 2. MDCSpell consists of a transformer-based detection network and a BERT-based correction network. These two networks use the same word embedding as input. At the end of the correction network, the hidden states from the correction and detection networks are fused into the classification dense layer as input to generate the correction results. These two tasks can be trained simultaneously in an end-to-end manner.

More specifically, we first generate the embedding required by BERT for each character, which is the sum of word embedding, position embedding, and segment embedding. Then we input the embedding sequence of the input text into the detection network and the correction network to obtain the encoded vector respectively. The detection network is a structure based on a multi-layer transformer, which needs to fit whether the characters in each position are misspelled. Therefore, the output encoded vector of the detection network contains the information of the possible error probability of the position. The correction network is a structure based on BERT, which needs to detect what characters need to be output in each position. Next, we fuse the information of the two encoded vectors to generate the final encoded vector. Lastly, a dense layer initialized by the transpose of the word embedding table takes the final encoded vector as input and generates the prediction result.

3.4 Detection Network

The detection network is a binary classification task based on the transformer structure, which is used to determine the error probability of characters in each position. For input text of length n, the input of detection network is the embedding sequence $E = (e_1, e_2, ..., e_n)$ of characters, which is the sum of word embedding, position embedding, and segment embedding. Then a context encoder is used to get the detection encoding vector. Finally, a projection layer is used to project the encoding vector into two-dimensional space, which represents the probability of correctness and error of the position character respectively.

Specifically, in order to capture better context semantics, we use a multi-layer transformer for encoding, where each layer uses the same block structure. The definition of each transformer block is as follows:

$$MultiHead = Concat(head_1, ..., head_n)W^O \quad (1)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

$$FFN(X) = max(0, XW_1 + b_1)W_2 + b_2 \quad (3)$$

Where Q, K, and V represent the representation of the current input sequence, which could be the embedding of characters or the output of the previous transformer block. $MultiHead$ and FFN represent multi-head self-attention and feed-forward network respectively, which are the basic components of the transformer. We denote the sequence of hidden states at the last layer of transformer blocks as $H^d = (h_1^d, h_2^d, ..., h_n^d)$.

The hidden state H^d is both used to predict the positions of the misspelled characters and deliver the position information to the correction network. Specifically, we use a dense layer as the output layer and the softmax function 4 to determine if an error happens. For each character of the raw input, the probability of error detection is defined as

$$P^d(g_i = 1|X) = \sigma(Wh_i^d + b) \tag{4}$$

where $P^d(g_i = 1|X)$ is the conditional probability which represents how likely the character corresponding to h_i^d is misspelled, σ represents the nonlinear function which we used sigmoid function, h_i^d denotes the final layer of the transformer-based detection network, W and b are the parameters of the dense layer.

3.5 Correction Network

Correction network is a multi-class classification task based on BERT-base, which is used to find the correct characters to replace the misspelled characters. BERT-base is composed of a stack of 12 identical transformer blocks. We denote the sequence of hidden states at the final layer of transformer blocks as $H^c = \{h_1^c, h_2^c, ..., h_n^c\}$.

Then we fuse the hidden states from the detection network and the correction network. In this work, we specially set the dimensions of the last hidden layer of the two networks to be the same, so we directly add them to get the fused representation

$$H = H^d + H^c \tag{5}$$

where H^d is the hidden states from the final layer of the transformer-based detection network and H^c is the hidden states from the final layer of the BERT-based correction network.

Lastly, we reviewed the correction task and we do not regard the correction task as a classification task through a random initialization projection layer, but as a similarity task, i.e., if the character of one position is correct, then through the encoding of detection and correction network, the final encoded vector should be very similar to the word embedding of the input character. On the contrary, if the character of one position is wrong, then the final encoded vector should be similar to the word embedding of the corrected character. The formula for the classification task is as follows.

$$P(y_i|X) = softmax(Wh_i) \tag{6}$$

Specifically, we use the transpose of the word embedding table to initialize the weight of projection layer W instead of random initialization. The result is that the large number of randomly initialized parameters of the projection matrix, could lead to slow convergence, and finally lead to poor performance. Instead, we use the transpose of the word embedding table to initialize the weights of the projection layer considering their similarity. By doing so, the training of the correction network converges much faster and steadily achieves desired performance.

3.6 Training

We define the detection task as the classification task of whether the character should be modified, and the correction task as the classification task of what the correct character is and formalize their loss functions as

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

$$L^c = -\sum_{i=1}^{n} logP^c(y_i|X) \tag{8}$$

where L^d and L^c are the loss functions for the training of the detection network and correction network respectively. Finally, we linearly combine the two functions as the overall loss function,

$$L = \lambda L^c + (1 - \lambda)L^d \tag{9}$$

where $\lambda \in [0, 1]$ is the coefficient to balance the detection loss and correction loss. We then simultaneously train the whole network by minimizing the L.

4 Experimental Results

4.1 Datasets

The training data is composed of three training datasets (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015), which has 10K data samples in total. Following (Wang et al., 2019), we also include

Training Data	#Line	Avg.Length	#Errors
(Wang et al., 2019)	271,329	44.4	382,704
SIGHAN 2013	350	49.2	350
SIGHAN 2014	6,526	49.7	10,087
SIGHAN 2015	3,174	30.0	4,237
Test Data	#Line	Avg.Length	#Errors
SIGHAN 2013	1,000	74.1	996
SIGHAN 2014	1,062	50.1	529
SIGHAN 2015	1,100	30.5	550

Table 2: Statistics information of the used data resources. The number in the bracket in #Line column denotes the number of sentences with errors.

additional 271K samples as the training data, which are generated by an automatic method (Wang et al., 2018).

To evaluate the performance of the proposed method, we used three test datasets from the SIGHAN 2013, SIGHAN 2014, SIGHAN 2015 benchmarks(Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) as in (Wang et al., 2019). We also follow the same data pre-processing procedure, i.e., the characters in these datasets are converted to simplified Chinese using OpenCC. The statistic of the data is listed in Table 2.

4.2 Baselines

We compare our method with the following typical baselines.

- Hybrid (Wang et al., 2018): This method uses a BiLSTM-based model trained on a generated dataset.
- FASpell (Hong et al., 2019): This method utilizes a specialized candidate selection method based on the similarity metric. This metric is measured using some empirical methods, e.g., edit distance, rather than a pre-defined confusion set.
- BERT (Devlin et al., 2019):The word embedding is used as the softmax layer on the top of BERT for the CSC task. We trained this model using the same setting as our baseline model.
- Soft-Masked BERT (Zhang et al., 2020): This method uses a two-stage detection and correction pipeline method, it masked the detected error character and then turn the input into the BERT model for error correction.
- SpellGCN (Cheng et al., 2020): This method incorporates phonological and visual similarity knowledge into BERT via a specialized graph convolutional network.
- GAD (Guo et al., 2021): This method learns the global relationship of the potential correct input characters and the candidates of potential error characters.
- DCN (Wang et al., 2021): This method uses a dynamic connected network to model the dependencies between two adjacent Chinese characters.

4.3 Evaluation Metrics

The sentence-level precision, recall, and F1 score are reported as the evaluation metrics as in most of the previous work. These metrics are provided for the detection and correction sub-tasks. We consider a sentence to be correctly annotated only if all errors in the sentence are corrected as in (Hong et al., 2019).

4.4 Training Details

We use the pretrained BERT as the correction network. For the sake of faster convergence, we initialize the weights of the transformer in the detection module with the first two layers and the embedding layer of BERT. The overall training process is divided into two stages for training. The first stage is to use nearly all 3 million training data to fine-tune the model, where the batch size is 32 and the learning rate is 2e-5. The second stage is to fine-tune the model on the SIGHAN training data, where the batch size is 32 and the learning rate is 1e-5.

4.5 Main Results

The main results can be found in Table 3. According to this table, our proposed MDCSpell framework consistently achieves the best F1 score, both for the detection task and the correction task, on all of the three datasets. The MDCSpell with the best model setup achieves 0.2%, 0.8%, 2.0% absolute gains on the three datasets compared to the best CSC method, indicating the effectiveness of our method. Also note compared with the BERT baseline (which is the correction part of our MDCSpell), our methods significantly improves the correction F1 score by 7.2%, 4.0%, 5.3% respectively, which illustrates the effectiveness of the detection network in the proposed multi-task architecture.

We can also find that the precision results significantly outperforms the competitors. Compared

Dataset	Model	Detection			Correction		
		Prec.	Rec.	F1.	Prec.	Rec.	F1.
SIGHAN 13	Hybrid (Wang et al., 2018)	54.0	69.3	60.7	-	-	52.1
	FASpell (Hong et al., 2019)	76.2	63.2	69.1	73.1	60.5	66.2
	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
	BERT(baseline)	79.0	72.8	75.8	77.7	71.6	74.6
	MDCSpell(ours)	**89.1**	78.3	**83.4**	**87.5**	76.8	**81.8**
SIGHAN 14	Hybrid (Wang et al., 2018)	51.9	66.2	58.2	-	-	56.1
	FASpell (Hong et al., 2019)	61.0	53.5	57.0	59.4	52.0	55.4
	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
	BERT(baseline)	65.6	68.1	66.8	63.1	65.5	64.3
	MDCSpell(ours)	**70.2**	68.8	**69.5**	**69.0**	67.7	**68.3**
SIGHAN 15	Hybrid (Wang et al., 2018)	56.6	69.4	62.3	-	-	57.1
	FASpell (Hong et al., 2019)	67.6	60.0	63.5	66.6	59.1	62.6
	Soft-Masked BERT (Zhang et al., 2020)	73.7	73.2	73.5	66.7	66.2	66.4
	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
	BERT(baseline)	73.7	78.2	75.9	70.9	75.2	73.0
	MDCSpell(ours)	**80.8**	80.6	**80.7**	**78.4**	**78.2**	**78.3**

Table 3: Experimental results of sentence-level precision, recall, and F1 score (%).

with the best competitor, our method has increased the precision by 2.8%, 3.2%, 3.9% on three datasets respectively. This improvement mainly benefits from the better usage of the detection information which aims to avoid the errors caused by the misleading impact on the context from the misspelled characters as well as make use of the visual and phonological features from them.

It is worth noting that although our method has achieved overall optimal results on precision and F1 score, the recall has a certain gap compared to some methods on these datasets. The reason might be that we did not use any external knowledge like confusion set compared to GAD and DCN. The competitive results achieved by us without using the external knowledge also illustrate the effectiveness of our method.

4.6 Ablation Study

In this subsection, we analyze the effect of the hyperparameters, including the number of detection layers and the value of λ. We evaluate their influence on the SIGHAN15 dataset.

Figure 3 shows the effect on the number of transformer layers in detection network and with or without BERT weights initialization. We compared the effect of the number of transformer layers from 0 to 4 based on the detection F1 score. From the figure we can find that, 1) the results of using BERT weights initialization greatly outperforms that of random initialization no matter how many transformer layers (> 0) are used, 2) as for the number of transformer layers, the best trade-off between performance and number of parameters can be achieved when the number of layers is 2 when using BERT weights initialization. In the main experiment, we used two transformer layers as the detection network with the BERT weights initialization.

In the multi-task learning, the impact of the selection of the scale parameter λ in the loss function on the result is shown in Figure 4. From this result, we can find that setting λ as 0.85 achieves the overall best correction F1 score. This is reasonable since the convergence of the correction task is harder than that of the detection task so that it demands a higher weight during learning. Meanwhile, an excessive high λ would diminish the learning

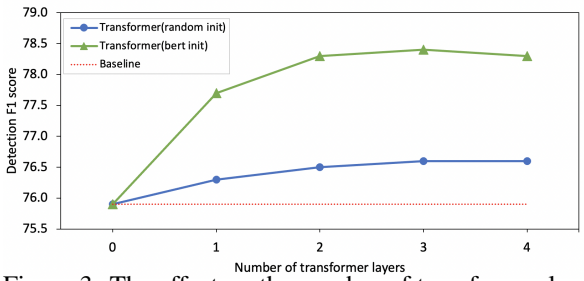


Figure 3: The effect on the number of transformer layers in detection network and with or without BERT weights initialization.

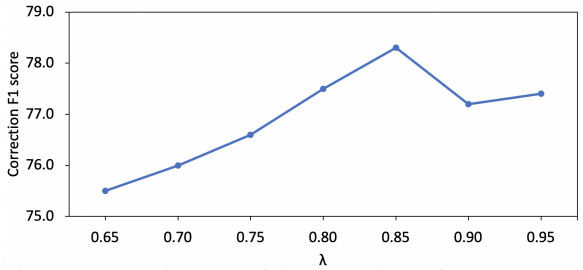


Figure 4: The impact of the selection of the scale parameter λ in the loss function.

of detection which might reduce the contribution from the detection network. Thus a relative higher λ achieves the overall best balance between the learning of these two tasks.

4.7 Case Study

To further analyze our approach, we show several correction results in Table 4 to demonstrate the properties of MDCSpell. It can be seen from the examples that MDCSpell can well capture the context and make judgments without being disturbed by the context of the wrong characters, and correct the wrong characters to its corrected counterparts. In the first case, that MDCSpell does not mistakenly change the correct token "哪里" in the context to the wrong token "那里" that often appears in the corpus like the baseline. In the second case, when there are multiple words that need to be corrected, MDCSpell successfully avoids the misleading affect from the context of the wrong characters, and correct multiple consecutive wrong characters "纳福境" into the correct characters "那附近" to maintain the fluent semantics. Also, it can be seen from the third case that the baseline mistakenly changes the character "高" into "寒" which is apparently affected by the original wrong character "心" since "寒心" is a meaningful word compared to "高心" in Chinese. On the contrary, MDCSpell can avoid this negative impact and successfully change the wrong characters into the correct ones. The success in solving these cases also proves the effectiveness of the MDCSpell.

Input	哪里(nǎ lǐ)有上大学，不想念书的道理?
baseline	那里(nà lǐ)有上大学，不想念书的道理?
MDCSpell	哪里(nǎ lǐ)有上大学，不想念书的道理?
Translation	What is the reason to go to university and not want to study?
Input	从那里，我们可以走到纳福境(nà fú jìng)的新光三钺百货公司逛一逛
baseline	从那里，我们可以走到纳福境(nà fú jìng)的新光三钺百货公司逛一逛
MDCSpell	从那里，我们可以走到那附近(nà fù jìn)的新光三钺百货公司逛一逛
Translation	From there, we can walk to the Shinkong Sanyue Department Store nearby.
Input	他主动拉了姑娘的手，心里很高心(gāo xīn)，嘴上故作生气
baseline	他主动拉了姑娘的手，心里很寒心(hán xīn)，嘴上故作生气
MDCSpell	他主动拉了姑娘的手，心里很高兴(gāo xìng)，嘴上故作生气
Translation	He took the girl's hand on his own initiative, very happy in his heart, pretending to be angry.

Table 4: Examples of CSC results, the incorrect and correct characters marked in red and blue respectively.

5 Conclusions

Spelling errors have two sides to the CSC task. Specifically, their visual and phonological features are critical for substitution for the correct characters, but their misleading impact on the context can mislead the correction model in turn. In this paper, we proposed a general detector-corrector multi-task framework MDCSpell which exploits the visual and phonological features of the misspelled characters and meanwhile minimizes their misleading impact on the context. The experiments demonstrate the effectiveness of our method. For future work, we will explore how to make better use of external knowledge to strengthen our model and other ways of using the detection information and extend the proposed framework to other problems like grammatical error correction.

References

Haithem Afli, Zhengwei Qiu, Andy Way, and Páraic Sheridan. 2016. Using smt for ocr error correction of historical texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 962–966.

Jill Burstein and Martin Chodorow. 1999. Automated essay scoring for nonnative english speakers. In

Computer mediated language assessment and evaluation in 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.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

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.

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.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065, Melbourne, Australia. Association for Computational Linguistics.

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.

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.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762, Vancouver, Canada. Association for Computational Linguistics.

Jing Li, 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.

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.

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.

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.

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.

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.

Weijian Xie, Peijie Huang, Xinrui Zhang, Kaiduo Hong, Qiang Huang, Bingzhou Chen, and Lei Huang. 2015. Chinese spelling check system based on n-gram model. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 128–136, Beijing, China. Association for Computational Linguistics.

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.

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, Wuhan, China. Association for Computational Linguistics.

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.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked bert.