Improving Chinese Spelling Check by Character Pronunciation Prediction: The Effects of Adaptivity and Granularity

Jiahao Li[1], Quan Wang[2*], Zhendong Mao[1], Junbo Guo[3], Yanyan Yang[4], Yongdong Zhang[1]

[1]University of Science and Technology of China, Hefei, China
[2]MOE Key Laboratory of Trustworthy Distributed Computing and Service, Beijing University of Posts and Telecommunications, Beijing, China
[3]People's Daily Online Co., Beijing, China
[4]People's Public Security University of China, Beijing, China
jiahao66@mail.ustc.edu.cn, wangquan@bupt.edu.cn, zdmao@ustc.edu.cn
guojunbo@people.cn, zhyd73@ustc.edu.cn

Abstract

Chinese spelling check (CSC) is a fundamental NLP task that detects and corrects spelling errors in Chinese texts. As most of these spelling errors are caused by phonetic similarity, effectively modeling the pronunciation of Chinese characters is a key factor for CSC. In this paper, we consider introducing an auxiliary task of Chinese pronunciation prediction (CPP) to improve CSC, and, for the first time, systematically discuss the adaptivity and granularity of this auxiliary task. We propose SCOPE which builds on top of a shared encoder two parallel decoders, one for the primary CSC task and the other for a fine-grained auxiliary CPP task, with a novel adaptive weighting scheme to balance the two tasks. In addition, we design a delicate iterative correction strategy for further improvements during inference. Empirical evaluation shows that SCOPE achieves new state-of-the-art on three CSC benchmarks, demonstrating the effectiveness and superiority of the auxiliary CPP task. Comprehensive ablation studies further verify the positive effects of adaptivity and granularity of the task. Code and data used in this paper are publicly available at https://github.com/jiahaozhenbang/SCOPE.

1 Introduction

Chinese Spelling Check (CSC), which aims to detect and correct spelling errors in Chinese texts, is a fundamental task in Chinese natural language processing. Spelling errors mainly originate from human writing errors and machine recognition errors, *e.g.*, errors caused by automatic speech recognition (ASR) and optical character recognition (OCR) systems (Huang et al., 2021). With the latest development of deep neural networks, neural CSC methods, in particular those based on encoder-decoder architectures, have become the mainstream of research in recent years (Xu et al., 2021; Liu et al., 2021). Encoder-decoder models regard CSC as a special sequence-to-sequence (Seq2Seq) problem, where a sentence with spelling errors is given as the input and a corrected sentence of the same length will be generated as the output.

Instance	Similarity	
	Coarse	Fine
W: 我觉得你们会好好的完(wan2/w,an,2)。 I think you will finish well. R: 我觉得你们会好好的玩(wan2/w,an,2)。 I think you will play well.	1	1
W: 我以前想要高(gao1/g,ao,1)诉你。 I tried to high you before. R: 我以前想要告(gao4/g,ao,4)诉你。 I tried to tell you before.	0	2/3
W: 他收(shou1/sh,ou,1)到山上的时候。 When he received the mountain. R: 他走(zou3/z,ou,3)到山上的时候。 When he walked up the mountain.	0	1/3
W: 行为都被蓝(lan2/l,an,2)控设备录影。 Actions are recorded by blue control devices. R: 行为都被监(jian1/j,ian,1)控设备录影。 Actions are recorded by surveillance devices.	0	0

Table 1: Instances from SIGHAN15 (Tseng et al., 2015). For each instance, coarse-/fine-grained pinyin of the misspelled (red) and correct (blue) characters are provided, along with their phonological similarity degree (the fraction of identical components) in terms of these two types of pinyin.

Previous research has shown that about 76% of Chinese spelling errors are induced by phonological similarity (Liu et al., 2011). Hence, it is a crucial factor to effectively model the pronunciation of Chinese characters for the CSC task. In fact, almost

*Corresponding author: Quan Wang.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4275–4286
December 7-11, 2022

all current advanced CSC approaches have actually exploited, either explicitly or implicitly, character pronunciation. The implicit use takes into account phonological similarities between pairs of characters, *e.g.*, by increasing the decoding probability of characters with similar pronunciation (Cheng et al., 2020) or integrating such similarities into the encoding process via graph convolutional networks (GCNs) (Cheng et al., 2020). The explicit use considers directly the pronunciation, or more specifically, pinyin[1], of individual characters, encoding the pinyin of input characters to produce extra phonetic features (Xu et al., 2021; Huang et al., 2021) or decoding the pinyin of target correct characters to serve as an auxiliary prediction task (Liu et al., 2021; Ji et al., 2021).

This paper also considers improving CSC with auxiliary character pronunciation prediction (CPP), but focuses specifically on the *adaptivity* and *granularity* of the auxiliary task, which have never been systematically studied before. First, all the prior attempts in similar spirit simply assigned a universal trade-off between the primary and auxiliary tasks for all instances during training, while ignoring the fact that the auxiliary task might provide different levels of benefits given different instances. Take for example the instances shown in Table 1. Compared to the misspelled character "蓝" and its correction "监" in the 4th instance, the two characters "完" and "玩" in the 1st instance are much more similar in pronunciation, suggesting that the spelling error there is more likely to be caused by phonological similarity, to which the pronunciation-related auxiliary task might provide greater benefits and hence should be assigned a larger weight. Second, prior efforts mainly explored predicting the whole pinyin of a character, *e.g.*, "gao1" for "高". Nevertheless, a syllable in Chinese is inherently composed of an initial, a final, and a tone, *e.g.*, "g", "ao", and "1" for "高". This fine-grained phonetic representation can better reflect not only the intrinsic regularities of Chinese pronunciation, but also the phonological similarities between Chinese characters. Consider for example the "高" and "告" case from the 2nd instance in Table 1. These two characters show no similarity in terms of their whole pinyin, but actually they share the same initial and final, differing solely in their tones.

Based on the above intuitions we devise **SCOPE** (*i.e.*, **S**pelling **C**heck by pr**O**nunciation **PrE**diction), which introduces a fine-grained CPP task with an adaptive task weighting scheme to improve CSC. Figure 1 provides an overview of SCOPE. Given a sentence with spelling errors as input, we encode it using ChineseBERT (Sun et al., 2021) to produce semantic and phonetic features. Then we build on top of the encoder two parallel decoders, one to generate target correct characters, *i.e.*, the primary CSC task, and the other to predict the initial, final and tone of the pinyin of each target character, *i.e.*, the auxiliary fine-grained CPP task. The trade-off between the two tasks can be further adjusted adaptively for each instance, according to the phonological similarity between input and target characters therein. In addition, we design an iterative correction strategy during inference to address the over-correction issue and tackle difficult instances with consecutive errors.

We empirically evaluate SCOPE on three shared benchmarks, and achieve substantial and consistent improvements over previous state-of-the-art on all three benchmarks, demonstrating the effectiveness and superiority of our auxiliary CPP task. Comprehensive ablation studies further verify the positive effects of adaptivity and granularity of the task.

The main contributions of this paper are summarized as follows: (1) We investigate the possibility of introducing an auxiliary CPP task to improve CSC and, for the first time, systematically discuss the adaptivity and granularity of this auxiliary task. (2) We propose SCOPE, which builds two parallel decoders upon a shared encoder for CSC and CPP, with a novel adaptive weighting scheme to balance the two tasks. (3) We establish new state-of-the-art on three benchmarking CSC datasets.

2 Related Work

CSC is a fundamental NLP task that has received wide attention over the past decades. Early work on this topic was mainly based on manually designed rules (Mangu and Brill, 1997; Jiang et al., 2012). After that, statistical language models became the mainstream for CSC (Chen et al., 2013; Yu and Li, 2014; Tseng et al., 2015). Methods of this kind in general followed a pipeline of error detection, candidate generation, and candidate selection. Given a sentence, the error positions are first detected by the perplexity of a language model. The candidates for corrections can then be generated according to similarity between characters, typically by using

[1]Pinyin is the official phonetic system of Mandarin Chinese, which literally means "spelled sounds".

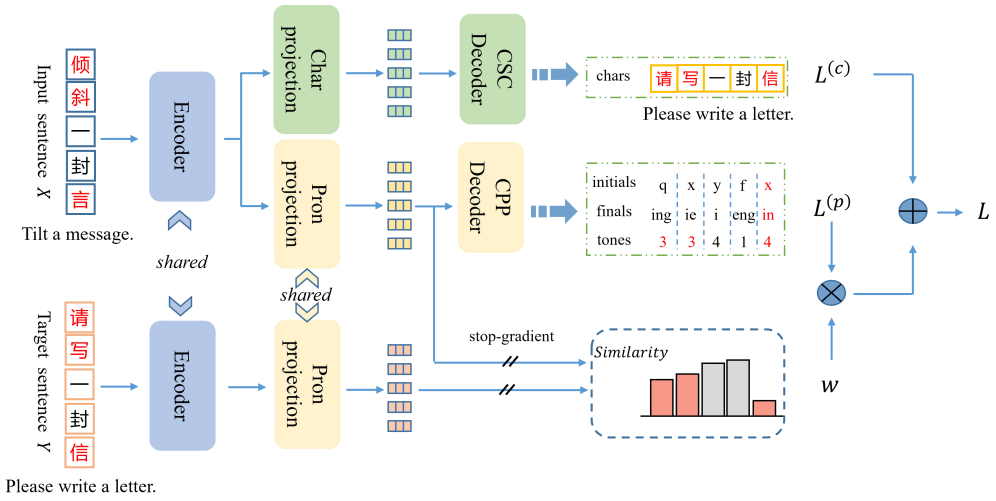

Figure 1: Overview of SCOPE. **Top:** The one-encoder-two-decoder structure for CSC and CPP. The input sentence X is fed into the encoder and then, after character-/pronunciation-specific feature projection, two parallel decoders, one to predict the characters, the other to predict the initial, final, and tone of each character in the target sentence. **Bottom:** Adaptive task weighting between CSC and CPP (detached in the backward pass). The target sentence Y is fed into the encoder and the pronunciation-specific feature projection layer. Then the similarities between input and target sentences on character level are calculated and the adaptive weights are accordingly defined. **Note:** Only the CSC decoder branch (along with the encoder) will be used at inference time.

a confusion set. And the final corrections can be determined by scoring the sentence replaced by the candidates with the language model (Liu et al., 2013; Xie et al., 2015).

In the era of deep learning, especially after Transformer (Vaswani et al., 2017) and pre-trained language models like BERT (Devlin et al., 2019) were proposed, a large number of neural CSC methods have emerged. Hong et al. (2019) used Transformer as an encoder to produce candidates and designed a confidence-similarity decoder to filter these candidates. Zhang et al. (2020) designed a detection network based on Bi-GRU to predict the error probability of each character and passed the probabilities to a BERT-based correction network via a soft masking mechanism. Cheng et al. (2020) employed GCNs combined with BERT to further model interdependences between characters. Recent work of (Xu et al., 2021; Liu et al., 2021; Huang et al., 2021) proposed to encode phonetic and glyph information in addition to semantic information, and then combine phonetic, glyph and semantic features to make final predictions.

As we could see, modeling pronunciation information is prevailing in CSC research (Zhang et al., 2021), typically via an encoding process to extract phonetic features. Liu et al. (2021) proposed the first work that considered predicting the pronunciation of target characters as an auxiliary task. Their work, however, employed pronunciation prediction in a coarse-grained, non-adaptive manner, which is quite different to ours.

3 Our Approach

This section presents our approach SCOPE for the CSC task. Below, we first define the problem formulation and then describe our approach in detail.

3.1 Problem Formulation

The Chinese spelling check (CSC) task is to detect and correct spelling errors in Chinese texts. Given a misspelled sentence $X = \{x_1, x_2, \cdots, x_n\}$ with n characters, a CSC model takes X as input, detects potential spelling errors on character level, and outputs a corresponding correct sentence $Y = \{y_1, y_2, \cdots, y_n\}$ of equal length. This task can be viewed as a conditional sequence generation problem that models the probability of $p(Y|X)$. We are further given the fine-grained pinyin of each character y_i in the correct sentence Y, represented as a triplet in the form of $(\alpha_i, \beta_i, \gamma_i)$, where α_i, β_i, and γ_i indicate the initial, final, and tone, respectively. Note that such kind of pinyin of the output sentence is required and provided solely during training.[2]

[2]In fact, we also use the pinyin of each character x_i in the input sentence X during the ChineseBERT encoding process (detailed later), and this kind of pinyin of the input sentence is required and provided during both training and inference.

3.2 SCOPE Architecture

The key idea of SCOPE is to employ a fine-grained character pronunciation prediction (CPP) task with an adaptive task weighting scheme to improve CSC. In achieving this SCOPE builds upon a shared encoder two parallel decoders, one for the primary CSC task and the other for the auxiliary CPP task. The trade-off between the two tasks is further determined adaptively based on the phonological similarity between input and target characters. Figure 1 summarizes the overall architecture of SCOPE.

Encoder Similar to recent CSC approaches that leverage multimodal information (Liu et al., 2021; Xu et al., 2021), we use ChineseBERT (Sun et al., 2021) as the encoder to extract semantic, phonetic, and morphologic features as well for the CSC task. ChineseBERT is a pre-trained language model that incorporates both the pinyin and glyph information of Chinese characters. Specifically, for each character x_i in the input sentence X, the encoder first produces its char embedding, pinyin embedding, and glyph embedding, all with embedding size D. These three embeddings are then concatenated and mapped to a D-dimensional fused embedding via a fully connected layer. After that, just like in most other pre-trained language models, the fused embedding is added with a position embedding, and fed into a stack of successive Transformer layers to generate a contextualized representation $\boldsymbol{h}_i \in \mathbb{R}^D$ for the input character x_i. We denote the character representations after this encoding process as $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n\}$. As the encoder is not the main concern of this paper, we just provide a brief sketch of the encoder and refer readers to (Vaswani et al., 2017; Sun et al., 2021) for details.

Decoder for CSC This decoder is to predict the characters in the correct sentence Y based on the encoding output $\boldsymbol{H}$. Specifically, given each input character x_i, we first project its encoding output $\boldsymbol{h}_i$ into a character-specific feature space:

$$\boldsymbol{h}_i^{(c)} = \text{GeLU}\left(\boldsymbol{W}^{(c)}\boldsymbol{h}_i + \boldsymbol{b}^{(c)}\right), \quad (1)$$

and then predict the corresponding correct character $\hat{y}_i$ based on the projection output:

$$p(\hat{y}_i|X) = \text{softmax}\left(\boldsymbol{W}^{(y)}\boldsymbol{h}_i^{(c)} + \boldsymbol{b}^{(y)}\right). \quad (2)$$

Here $\boldsymbol{W}^{(c)} \in \mathbb{R}^{D\times D}$, $\boldsymbol{b}^{(c)} \in \mathbb{R}^D$ are learnable parameters of the character-specific feature projection layer; $\boldsymbol{W}^{(y)} \in \mathbb{R}^{V\times D}$, $\boldsymbol{b}^{(y)} \in \mathbb{R}^V$ are learnable parameters of the character prediction layer; V is the vocabulary size.

Decoder for CPP This decoder is to predict the fine-grained pinyin, *i.e.*, the initial, final, and tone, of each character in the correct sentence Y based on the encoding output $\boldsymbol{H}$. Again, given each input character x_i and its encoding output $\boldsymbol{h}_i$, we project $\boldsymbol{h}_i$ into a pronunciation-specific feature space:

$$\boldsymbol{h}_i^{(p)} = \text{GeLU}\left(\boldsymbol{W}^{(p)}\boldsymbol{h}_i + \boldsymbol{b}^{(p)}\right), \quad (3)$$

and predict the initial $\hat{\alpha}_i$, final $\hat{\beta}_i$, and tone $\hat{\gamma}_i$ of the corresponding correct character based on the projection output:

$$p(\hat{\alpha}_i|X) = \text{softmax}\left(\boldsymbol{W}^{(\alpha)}\boldsymbol{h}_i^{(p)} + \boldsymbol{b}^{(\alpha)}\right), \quad (4)$$

$$p(\hat{\beta}_i|X) = \text{softmax}\left(\boldsymbol{W}^{(\beta)}\boldsymbol{h}_i^{(p)} + \boldsymbol{b}^{(\beta)}\right), \quad (5)$$

$$p(\hat{\gamma}_i|X) = \text{softmax}\left(\boldsymbol{W}^{(\gamma)}\boldsymbol{h}_i^{(p)} + \boldsymbol{b}^{(\gamma)}\right). \quad (6)$$

Here $\boldsymbol{W}^{(p)} \in \mathbb{R}^{D\times D}$ and $\boldsymbol{b}^{(p)} \in \mathbb{R}^D$ are learnable parameters of the pronunciation-specific feature projection layer; $\boldsymbol{W}^{(\delta)} \in \mathbb{R}^{U\times D}$, $\boldsymbol{b}^{(\delta)} \in \mathbb{R}^U$ with $\delta \in \{\alpha, \beta, \gamma\}$ are learnable parameters of the pronunciation prediction layers; U is the total number of pronunciation units (initials, finals, and tones).

Adaptive Task Weighting We devise an adaptive task weighting scheme to balance the primary CSC and auxiliary CPP tasks during training. Given an input sentence X, the CSC task aims to match the predicted characters $\{\hat{y}_i\}_{i=1}^n$ with the ground truth $\{y_i\}_{i=1}^n$, while the CPP task aims to match the predicted fine-grained pinyin $\{(\hat{\alpha}_i, \hat{\beta}_i, \hat{\gamma}_i)\}_{i=1}^n$ with the ground truth $\{(\alpha_i, \beta_i, \gamma_i)\}_{i=1}^n$. Their loss functions are respectively defined as:

$$\mathcal{L}_i^{(c)} = -\log p(\hat{y}_i = y_i|X), \quad (7)$$

$$\mathcal{L}_i^{(p)} = -\frac{1}{3}\sum_{\delta\in\{\alpha,\beta,\gamma\}} \log p(\hat{\delta}_i = \delta_i|X), \quad (8)$$

where $\mathcal{L}_i^{(c)}$, $\mathcal{L}_i^{(p)}$ are the character and pronunciation prediction losses associated with the i-th character in the sentence, and the pronunciation prediction loss $\mathcal{L}_i^{(p)}$ is averaged over the initial, final, and tone prediction.

Then as we have discussed earlier in the introduction, the auxiliary CPP task might provide different levels of benefits given different input characters. The more similar the input and target characters are

in their pronunciation, the more likely there would be a spelling error caused by phonetic similarity. And to this case the CPP task might provide greater benefits and should be assigned a larger weight. To calculate such adaptive weights, we feed the target correct sentence Y to the encoder and the followup pronunciation-specific projection layer. Then we calculate for each input character x_i and its target character y_i a cosine similarity $\cos(\boldsymbol{h}_{x_i}^{(p)}, \boldsymbol{h}_{y_i}^{(p)})$ based on their pronunciation-specific feature representations $\boldsymbol{h}_{x_i}^{(p)}$, $\boldsymbol{h}_{y_i}^{(p)}$ (see Eq. (3)), and accordingly define the adaptive weight at the i-th position as:

$$w_i = e^{-(\cos(\boldsymbol{h}_{x_i}^{(p)}, \boldsymbol{h}_{y_i}^{(p)})-1)^2}. \tag{9}$$

The higher the cosine similarity $\cos(\boldsymbol{h}_{x_i}^{(p)}, \boldsymbol{h}_{y_i}^{(p)})$ is, the larger the weight w_i will be. Finally, the overall loss is defined as the CSC loss with an adaptively weighted CPP loss:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{L}_i^{(c)} + w_i\mathcal{L}_i^{(p)}\right), \tag{10}$$

where $\mathcal{L}_i^{(c)}$ and $\mathcal{L}_i^{(p)}$ are the character-specific CSC and CPP losses defined in Eq. (7) and Eq. (8), respectively. There are two points worth noting here: (1) The branch of encoding and mapping the target sentence Y is employed solely in the forward pass to calculate adaptive weights, and will be detached in the backward pass. (2) The auxiliary CPP task, as well as the adaptive weighting scheme, is introduced solely during training. At inference time, we use the CSC decoder alone for prediction.

3.3 Constrained Iterative Correction

As pointed out by Liu et al. (2022), advanced CSC models based on pre-trained language models (*e.g.*, BERT (Devlin et al., 2019) and ChineseBERT (Sun et al., 2021)) typically have poor performance on multi-typo texts, and tend to overcorrect valid expressions to more frequent expressions. To address these deficiencies, we devise a simple yet effective constrained iterative correction strategy during inference. Specifically, at inference time, for each input sentence we detect and correct spelling errors in an iterative fashion. During each iteration, only the corrections that appear in a specified window around each correction position in the previous iteration are allowed. After the iterations, if a position is modified every iteration, we restore this position to its original character without any correction. We empirically set the iteration number to 2 and the window size to 3 (*i.e.*, one position on the left and one on the right of the current position). As we will see later in our case study in Section 4.5, this iterative correction strategy can effectively address the overcorrection issue and tackle difficult instances with multiple, in particular, consecutive errors.

3.4 Further Pre-training with Confusion Set

To obtain better initialization for SCOPE, we perform further pre-training by using a confusion set, as commonly practiced in most recently proposed CSC models (Xu et al., 2021; Liu et al., 2021). We consider wiki2019zh[3] that consists of one million Chinese Wikipedia articles, split these articles into paragraphs, and regard each paragraph as a target sequence with no spelling errors. We further collect easily confused character pairs from a mixture of three publicly available confusion sets (Wu et al., 2013; Lee et al., 2019; Wang et al., 2018), and retain only the pairs where both characters appear frequently (top 40%) in the wiki2019zh corpus. Then, for each target sequence, we create a potentially misspelled input sequence by randomly selecting and replacing 15% of the characters. Each selected character is replaced with an easily confused character (if any) 80% of the time, a random character from the vocabulary 10% of the time, and remained unchanged 10% of the time. After that, we pre-train SCOPE on these misspelled and correct sequence pairs before adapting it to target datasets.

4 Experiments and Results

In this section, we introduce our experiments and results on SIGHAN benchmarks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). We then verify the effectiveness of our model design, in particular the adaptivity and granularity of the auxiliary CPP task, via extensive ablation studies and analyses.

4.1 Experimental Setups

Datasets and Evaluation Metrics As in previous work (Cheng et al., 2020; Liu et al., 2021; Xu et al., 2021), our training data is a combination of (1) manually annotated training examples from SIGHAN13 (Wu et al., 2013), SIGHAN14 (Yu et al., 2014), SIGHAN15 (Tseng et al., 2015), and (2) 271K training examples from Wang et al. (2018) automatically generated by ASR- and OCR-based methods. We employ the test sets of SIGHAN13,

[3] https://github.com/brightmart/nlp_chinese_corpus

Training Set	#Sent	Avg. Length	#Errors
SIGHAN15	2,338	31.1	3,037
SIGHAN14	3,437	49.6	5,122
SIGHAN13	700	41.8	343
Wang271K	271,329	42.6	381,962
Test Set	**#Sent**	**Avg. Length**	**#Errors**
SIGHAN15	1,100	30.6	703
SIGHAN14	1,062	50.0	771
SIGHAN13	1,000	74.3	1,224

Table 2: Statistics of the datasets, including the number of sentences, the average length of sentences in tokens, and the number of errors in characters. We train on a combination of the training sets, and evaluate separately on each test set.

SIGHAN14, SIGHAN15 for evaluation. The statistics of the used datasets are shown in Table 2. The original SIGHAN datasets are in traditional Chinese. We follow previous work (Cheng et al., 2020; Xu et al., 2021) to convert them to simplified Chinese using OpenCC[4]. We further use pypinyin[5] to obtain the pinyin of each character, and segment it into the initial, final, and tone using a pre-defined vocabulary of initials and finals provided by Xu et al. (2021).[6]

We use the widely adopted sentence-level precision, recall and F1 as our main evaluation metrics. Sentence-level metrics are stricter than character-level metrics since a sentence is considered to be correct if and only if all errors in the sentence are successfully detected and corrected. Metrics are reported on the detection and correction sub-tasks. Besides sentence-level evaluation, we also consider character-level evaluation and the official SIGHAN evaluation. We leave their results to Appendix A

Baseline Methods We compare SCOPE against the following baseline methods. All these methods have employed character phonetic information in some manner, and represent current state-of-the-art on the SIGHAN benchmarks.

- *FASpell* (Hong et al., 2019) employs BERT to generate candidates for corrections and filters visually/phonologically irrelevant candidates by a confidence-similarity decoder.
- *SpellGCN* (Cheng et al., 2020) learns pronunciation/shape similarities between characters via GCNs, and combines the graph representations with BERT output for final prediction.
- *MLM-phonetics* (Zhang et al., 2021) jointly fine-tunes a detection module and a correction module on the basis of a pre-trained language model with phonetic features.
- *REALISE* (Xu et al., 2021) models semantic, phonetic and visual information of input characters, and selectively mixes information in these modalities to predict final corrections.
- *PLOME* (Liu et al., 2021) extracts phonetic and visual features of characters using GRU. It also predicts the pronunciation of target characters, but in a coarse-grained, non-adaptive manner.

Implementation Details In SCOPE, the encoder is initialized from ChineseBERT-base[7], while the decoders are randomly initialized. We then conduct further pre-training on wiki2019zh for 1 epoch with a batch size of 512 and a learning rate of 10^{-4}. The other hyperparameters are set to their default values as in ChineseBERT (Sun et al., 2021). During this pre-training stage, we do not use the adaptive task weighting scheme, and simply set the auxiliary CPP task weight to 1 for all characters for computational efficiency. After that, we fine-tune on the combined training set. We set the maximum sequence length to 192 and the learning rate to $5{\times}10^{-5}$. The optimal models on SIGHAN13/SIGHAN14/SIGHAN15 are obtained by training with batch sizes of 96/96/64 for 20/30/30 epochs, respectively. Other hyperparameters are again set to their default values as in ChineseBERT. All experiments are conducted on 2 GeForce RTX 3090 with 24G memory.

4.2 Main Results

Table 3 presents the sentence-level performance of SCOPE and its baseline methods on the test sets of SIGHAN13, SIGHAN14, and SIGHAN15. We can see that SCOPE consistently outperforms all the baselines on all the datasets in almost all metrics, verifying its effectiveness and superiority for CSC. The improvements, in most cases, are rather substantial, *e.g.*, +2.5/+2.9 detection/correction F1 on SIGHAN15 and +1.8/+1.4 detection/correction F1 on SIGHAN14. Note that on SIGHAN13, although the improvements over the best performing baseline REALISE are somehow limited, SCOPE still

[4] https://github.com/BYVoid/OpenCC
[5] https://pypi.org/project/pypinyin
[6] https://github.com/DaDaMrX/ReaLiSe
[7] https://github.com/ShannonAI/ChineseBert

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN15	FASpell (Hong et al., 2019)	67.6	60.0	63.5	66.6	59.1	62.6
	SpellGCN (Cheng et al., 2020)	74.8	80.7	77.7	72.1	77.7	75.9
	MLM-phonetics (Zhang et al., 2021)	77.5	83.1	80.2	74.9	80.2	77.5
	REALISE (Xu et al., 2021)	77.3	81.3	79.3	75.9	79.9	77.8
	PLOME (Liu et al., 2021)	77.4	81.5	79.4	75.3	79.3	77.2
	SCOPE (ours)	**81.1**	**84.3**	**82.7**	**79.2**	**82.3**	**80.7**
SIGHAN14	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
	MLM-phonetics (Zhang et al., 2021)	66.2	**73.8**	69.8	64.2	**73.8**	68.7
	REALISE (Xu et al., 2021)	67.8	71.5	69.6	66.3	70.0	68.1
	SCOPE (ours)	**70.1**	73.1	**71.6**	**68.6**	71.5	**70.1**
SIGHAN13	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
	MLM-phonetics (Zhang et al., 2021)	82.0	78.3	80.1	79.5	77.0	78.2
	REALISE (Xu et al., 2021)†	**88.6**	82.5	**85.4**	**87.2**	81.2	84.1
	SCOPE (ours)†	87.4	**83.4**	**85.4**	86.3	**82.4**	**84.3**

Table 3: Sentence-level performance on the test sets of SIGHAN13, SIGHAN14, SIGHAN15, where precision (P), recall (R), F1 (F) for detection (D) and correction (C) are reported (%). Baseline results are directly taken from their respective literatures. Results marked by "†" are obtained by applying a post-processing step on SIGHAN13 which removes all detected and corrected "的", "地", "得" from the model output before evaluation, due to the relatively poor annotation quality about "的", "地", "得" on SIGHAN13 as observed and suggested by Xu et al. (2021).

outperforms the second best performing baseline MLM-phonetics by large margins (+5.3/+6.1 detection/correction F1). We attribute this phenomenon to the fact that the annotation quality is relatively poor on SIGHAN13, with a lot of mixed usage of "的", "地", "得" not annotated (Cheng et al., 2020). We hence follow REALISE (Xu et al., 2021) and remove all detected and corrected "的", "地", "得" from the model output before evaluation. This post-processing trick is extremely useful on SIGHAN13, and it might even conceal improvements from other strategies on this dataset.

Besides sentence-level metrics, we also consider character-level evaluation and the official SIGHAN evaluation, and make further comparison to some other methods that have their results reported in these settings (Ji et al., 2021; Liu et al., 2022). We leave the results to Appendix A, which reveal that SCOPE still performs the best in these new settings.

4.3 Eliminating Encoder Differences

As SCOPE uses a different and potentially more powerful encoder (*i.e.*, ChineseBERT) compared to the baselines, we further conduct experiments to eliminate the effects of different encoders and focus solely on the auxiliary CPP task, which is the main contribution of this work. To do so, we initialize the encoder from a well-trained REALISE model (one of the best performing baselines with its code and model released to the public). Then, we perform further pre-training on wiki2019zh and fine-tune on the combination of SIGHAN benchmarks with our adaptively-weighted, fine-grained CPP task. The pre-training and fine-tuning configurations are the same as those introduced above in Section 4.1. The constrained iterative correction (CIC) strategy is also applied during inference. We call this setting SCOPE (REALISE).

	Detection-level			Correction-level		
	D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN15						
REALISE	77.3	81.3	79.3	75.9	79.9	77.8
SCOPE (REALISE)	**78.7**	**84.7**	**81.6**	**76.8**	**82.6**	**79.6**
SIGHAN14						
REALISE	67.8	71.5	69.6	66.3	70.0	68.1
SCOPE (REALISE)	**69.0**	**75.0**	**71.9**	**67.1**	**72.9**	**69.9**
SIGHAN13						
REALISE	**88.6**	82.5	**85.4**	**87.2**	81.2	84.1
SCOPE (REALISE)	87.5	**83.2**	85.3	86.4	**82.3**	**84.3**

Table 4: Performance of SCOPE with the same encoder as REALISE on test sets of SIGHAN13, SIGHAN14, and SIGHAN15.

Table 4 presents the sentence-level performance of this new setting on the test sets of SIGHAN13, SIGHAN14, and SIGHAN15. We can observe that SCOPE (REALISE) consistently outperforms its

direct opponent REALISE on all the datasets. The improvements, in most cases, are rather substantial, except for those on the relatively poorly annotated SIGHAN13. These results verify the effectiveness of our approach irrespective of the encoder.

4.4 Effects of Adaptivity and Granularity

This section then investigates the effects of *adaptivity* and *granularity* of the auxiliary CPP task on the overall CSC performance.

Adaptivity As for adaptivity, we make comparison among the following three diverse task weighting schemes that balance the CSC and CPP tasks.

- *Fully-adaptive* (Full-adapt) is the scheme we used in SCOPE. It determines the CPP task weights according to phonological similarities between input and target characters, and the similarities are further adjusted dynamically during model training (see Eq. (9)).
- *Partially-adaptive* (Part-adapt) also decides the CPP task weights according to phonological similarities, but the similarities are static, defined as $w_i = 1 - \mathrm{norm}(\mathrm{edit\_distance}_i)$, where $\mathrm{edit\_distance}_i$ is the Levenshtein edit distance (Levenshtein et al., 1966) between the pinyin sequences of the i input and target characters and $\mathrm{norm}(\cdot)$ is a normalization function. The smaller the edit distance is, the larger the weight will be.
- *Non-adaptive* (Non-adapt) considers no adaptivity and simply sets the CPP task weight to 1 for all characters ($w_i = 1$ for all i).

We compare the three settings in the SIGHAN fine-tuning stage, starting from the same checkpoint after pre-training on wiki2019zh with a non-adaptive task weighting scheme. Here Full-adapt is equivalent to SCOPE.

Granularity As for granularity, we consider and make comparison between two types of CPP tasks.

- *Fine-grained* (Fine) is the task we employed in SCOPE that predicts the initial, final, and tone of the pinyin of each target character.
- *Coarse-grained* (Coarse) is a task that predicts the whole pinyin of each target character.

For fair comparison, we also introduce further pre-training on wiki2019zh with a coarse-grained CPP task, and use this checkpoint to initialize the Coarse setting during SIGHAN fine-tuning. In both two settings the CPP task is adaptively weighted as in Eq. (9), and Fine is equivalent to SCOPE.

SIGHAN15	Detection-level			Correction-level		
	D-P	D-R	D-F	C-P	C-R	C-F
REALISE	77.3	81.3	79.3	75.9	79.9	77.8
w/o CPP	79.1	82.4	80.7	76.8	80.0	78.4
Effects of Adaptivity						
Non-adapt	79.0	83.5	81.2	76.6	81.0	78.7
Part-adapt	80.1	83.5	81.8	78.0	81.3	79.6
Full-adapt	81.1	84.3	82.7	79.2	82.3	80.7
Effects of Granularity						
Coarse	79.9	83.7	81.8	77.4	81.1	79.2
Fine	81.1	84.3	82.7	79.2	82.3	80.7

Table 5: Performance of SCOPE with different levels of adaptivity and granularity of the auxiliary CPP task on the test set of SIGHAN15.

Results Table 5 presents the sentence-level performance of these SCOPE variants on the test set of SIGHAN15. The scores of our best performing baseline REALISE as well as SCOPE without the CPP task (denoted as w/o CPP) are also provided for reference. We can see that introducing an auxiliary CPP task always brings benefits to CSC, no matter what level of adaptivity and granularity the task is. As for the adaptivity of task weighting, the Full-adapt scheme that considers dynamic adaptivity performs better than Part-adapt that considers static adaptivity, which in turn performs better than Non-adapt that considers no adaptivity. As for the granularity, a fine-grained CPP task performs better than a coarse-grained one. These results verify the rationality of introducing a fine-grained CPP task with adaptive task weighting to improve CSC.

4.5 Ablation and Case Study

Ablation Study We conduct ablation studies on SIGHAN15 with the following settings: (1) removing the auxiliary CPP task (w/o CPP); (2) removing further pre-training on wiki2019zh (w/o FPT); and (3) removing the constrained iterative correction strategy at inference time (w/o CIC). The results are presented in Table 6. We can see that no matter which component we remove, the performance of SCOPE drops. This fully demonstrates the effectiveness of each component in our method.

SIGHAN15	Detection-level			Correction-level		
	D-P	D-R	D-F	C-P	C-R	C-F
SCOPE	81.1	84.3	82.7	79.2	82.3	80.7
w/o CPP	79.1	82.4	80.7	76.8	80.0	78.4
w/o FPT	80.2	83.2	81.7	77.5	80.4	78.9
w/o CIC	78.3	82.6	80.4	76.5	80.8	78.6

Table 6: Ablation results on the test set of SIGHAN15. The following changes are applied to SCOPE: removing the CPP task (w/o CPP), removing further pre-training (w/o FPT), and removing constrained iterative correction (w/o CIC).

Case Study Table 7 further shows several cases from the SIGHAN15 test set to illustrate how the constrained iterative correction strategy (see Section 3.3) can effectively tackle consecutive spelling errors and address the over-correction issue. For consecutive errors, *e.g.*, "户 秃" in the first case, this strategy is able to correct them iteratively, one character at a time, *e.g.*, by modifying "秃" to "涂" in the first round and then "户" to "糊" in the second round. For over-correction where the model makes unnecessary modifications, *e.g.*, "他" to "她" in the third case and "隔" to "葛" in the fourth case, the iterative correction strategy can always change them back most of the time.

Tackle Consecutive Errors	
Input:	我以前想要高诉你，可我忘了。我真户秃。 I tried to high you before, but I forgot. I'm really house bald.
Iteration 1:	我以前想要告诉你，可我忘了。我真户涂。 I tried to tell you before, but I forgot. I'm really house painted.
Iteration 2:	我以前想要告诉你，可我忘了。我真糊涂。 I tried to tell you before, but I forgot. I'm really muddled.
Input:	可是福物生对我们很客气。 But the fortune object man was polite to us.
Iteration 1:	可是福务生对我们很客气。 But the fortune business man was polite to us.
Iteration 2:	可是服务生对我们很客气。 But the waiter was very polite to us.
Address Over-correction Issue	
Input:	他再也不会撤扬。 He will never withdraw raise again.
Iteration 1:	她再也不会撤样。 She will never withdraw appearance again.
Iteration 2:	他再也不会这样。 He will never do this again.
Input:	幸运地，隔天她带着辞典来学校。 Fortunately, she came to school the next day with a thesaurus.
Iteration 1:	幸运地，葛天她带着辞典来学校。 Fortunately, Ge Tian she came to school with a thesaurus.
Iteration 2:	幸运地，隔天她带着辞典来学校。 Fortunately, she came to school the next day with a thesaurus.

Table 7: Cases from the SIGHAN15 test set to show how the iterative correction strategy can tackle consecutive errors and address the over-correction issue. Erroneous characters are in red, and their SCOPE corrections are in blue and underlined.

5 Conclusions

This paper proposes SCOPE, which employs a fine-grained Chinese pronunciation prediction (CPP) task with adaptive task weighting to improve the performance of Chinese spelling check (CSC). Our method builds upon a shared encoder two parallel decoders, one to predict target characters *i.e.*, CSC, and the other to predict initials, finals, and tones of target characters, *i.e.*, fine-grained CPP. The two decoders are then balanced adaptively according to the phonetic similarity between input and target characters. An iterative correction strategy is further designed during inference. SCOPE establishes new state-of-the-art on three SIGHAN benchmarks, verifying the effectiveness and superiority of introducing an auxiliary CPP task to improve CSC. Extensive ablation studies further verify the positive effects of dynamic adaptivity and fine granularity of this auxiliary task.

Limitations

SCOPE introduces an auxiliary CPP task alongside the primary CSC task in the training phase. This auxiliary CPP task causes 28% extra overhead of computation, with the runtime per epoch increasing from 19.32 minutes to 24.68 minutes. But the extra overhead of GPU memory is almost negligible, as the CPP decoder contains only 1M out of the total 148M parameters of the whole model (to which the encoder contributes 146M parameters). Note that the additional overhead caused by CPP is required only in the training phase, but not at inference time.

Acknowledgements

We would like to thank all the reviewers for their insightful and valuable suggestions, which significantly improve the quality of this paper. This work is supported by National Natural Science Foundation of China under Grants 61876223, 62222212 and U19A2057, and Science Fund for Creative Research Groups under Grant 62121002.

References

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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 79–83. 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, ACL 2020, Online, July 5-10, 2020*, pages 871–881. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. 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, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1419–1428. Association for Computational Linguistics.

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@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 160–169. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5958–5967. 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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3544–3551. Association for Computational Linguistics.

Ying Jiang, Tong Wang, Tao Lin, Fangjie Wang, Wenting Cheng, Xiaofei Liu, Chenghui Wang, and Weijian Zhang. 2012. A rule based chinese spelling and grammar detection system utility. In *2012 International Conference on System Science and Engineering (ICSSE)*, pages 437–440. IEEE.

Lung Hao Lee, Wun Syuan Wu, Jian Hong Li, Yu Chi Lin, 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.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

C.-L. Liu, M.-H. Lai, Kan-Wen Tien, Y.-H. Chuang, Shih-Hung Wu, and C.-Y. Lee. 2011. Visually and phonologically similar characters in incorrect chinese words: Analyses, identification, and applications. *ACM Trans. Asian Lang. Inf. Process.*, 10(2):10:1–10:39.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2991–3000. 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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 54–58. Asian Federation of Natural Language Processing.

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 187–194. Morgan Kaufmann.

Zijun Sun, Xiaoya Li, Xiaofei Sun, Yuxian Meng, Xiang Ao, Qing He, Fei Wu, and Jiwei Li. 2021. Chinesebert: Chinese pretraining enhanced by glyph and pinyin information. In *Proceedings of the 59th Annual Meeting of the Association for Computational*

Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pages 2065–2075. Association for Computational Linguistics.

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, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 32–37. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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, Brussels, Belgium, October 31 - November 4, 2018*, pages 2517–2527. 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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 35–42. 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, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 128–136. Association for Computational Linguistics.

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, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 716–728. 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, Wuhan, China, October 20-21, 2014*, pages 220–223. 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, Wuhan, China, October 20-21, 2014*, pages 126–132. 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, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2250–2261. 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, ACL 2020, Online, July 5-10, 2020*, pages 882–890. Association for Computational Linguistics.

A Character-level and Official Evaluation

This section further compares SCOPE to some recently proposed methods that have not been evaluated with sentence-level metrics, but instead with character-level and/or official evaluation metrics. These baseline methods include:

- *SpellBERT* (Ji et al., 2021) uses a lightweight pre-trained model for CSC, encoding phonetic and visual features with GNNs.
- *GAD* (Guo et al., 2021) models the global dependency between all candidate characters by a global attention decoder.
- *CRASpell* (Liu et al., 2022) constructs a noise modeling module that makes their model robust to consecutive spelling errors, with a copy mechanism to handle over-correction.

For reference, we also include two previously compared baselines SpellGCN (Cheng et al., 2020) and PLOME (Liu et al., 2021) that have their results reported on these new metrics. We use the code released by REALISE (Xu et al., 2021)[8] for sentence-level evaluation and the code released by CRASpell (Liu et al., 2022)[9] for character-level evaluation. The official evaluation scripts are provided along with the datasets.[10,11,12]

[8] https://github.com/DaDaMrX/ReaLiSe
[9] https://github.com/liushulinle/CRASpell
[10] http://ir.itc.ntnu.edu.tw/lre/sighan7csc.html
[11] http://ir.itc.ntnu.edu.tw/lre/clp14csc.html
[12] http://ir.itc.ntnu.edu.tw/lre/sighan8csc.html

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN15	SpellGCN (Cheng et al., 2020)	77.7	85.6	81.4	96.9	82.9	89.4
	PLOME (Liu et al., 2021)	85.2	86.8	86.0	97.2	85.0	90.7
	CRASpell (Liu et al., 2022)	83.5	**89.2**	86.3	97.1	**86.6**	91.5
	SCOPE (ours)	**86.8**	88.9	**87.8**	**97.4**	**86.6**	**91.7**

Table 8: Character-level performance on the whole test set of SIGHAN15, with baseline results directly taken from their respective literatures.

Dataset	Model	Detection-level			Correction-level		
		D-P	D-R	D-F	C-P	C-R	C-F
SIGHAN15	SpellGCN (Cheng et al., 2020)	85.9	80.6	83.1	85.4	77.6	81.3
	PLOME (Liu et al., 2021)	87.9	80.9	84.3	87.6	78.3	82.7
	GAD (Guo et al., 2021)	86.0	80.4	83.1	85.6	77.8	81.5
	SpellBERT (Ji et al., 2021)	87.5	73.6	80.0	87.1	71.5	78.5
	SCOPE (ours)	**89.4**	**84.3**	**86.3**	**89.2**	**82.4**	**85.7**

Table 9: Official evaluation results on the whole test set of SIGHAN15, with baseline results directly taken from their respective literatures.

The results are shown in Table 8 and Table 9. We can see that regardless of the evaluation scenarios, SCOPE consistently outperforms all the baselines in almost all metrics, verifying its effectiveness and superiority for CSC.

B Hyperparameter Search

We conduct a hyperparameter search for learning rate, batch size and epoch. Learning rate is tuned from $\{2\times10^{-5}, 5\times10^{-5}\}$, batch size from $\{48, 64, 96\}$ and epoch from $\{20, 30\}$. There are 12 hyperparameter search trials in total on each dataset. The optimal configurations are given in Section 4.1.