DISC: Plug-and-Play Decoding Intervention with Similarity of Characters for Chinese Spelling Check

Anonymous EMNLP submission

Abstract

One of the key challenges in Chinese spelling check (CSC) is ensuring that modifications remain faithful to the original intent of the sentence. Confusion sets are commonly used to mitigate this issue; however, it is challenging to construct high-quality confusion sets and integrate them into the model. In this paper, we propose a plug-and-play DISC (Decoding Intervention with Similarity of Characters) module for CSC models to address these challenges. DISC measures phonetic and glyph similarities between characters and incorporates this similarity information in the decoding stage. This method can be easily integrated into various existing CSC models, such as ReaLiSe, SCOPE, and ReLM, without additional training costs. Experiments on three CSC benchmarks demonstrate that our proposed method significantly improves model performance, approaching and even surpassing the current state-of-the-art models.[1]

1 Introduction

Spelling errors can lead to ambiguity and degrade the performance of many natural language processing (NLP) tasks, such as named entity recognition (Yang et al., 2024) and search engine (Martins and Silva, 2004; Gao et al., 2010). Given a sentence, the goal of Chinese spelling check (CSC) is to correct all spelling errors, as shown in Table 1 (Bao et al., 2020; Xu et al., 2021; Liu et al., 2021; Wu et al., 2023; Dong et al., 2024).

One of the key challenges in CSC is ensuring that the modifications remain faithful to the original intent of the sentence. The most straightforward way to address this challenge is by leveraging a confusion set to identify characters that are easily confused with the original ones, and then selecting the most appropriate correction from these candidates (Yeh et al., 2013; Yu and Li, 2014; Huang et al., 2014; Xie et al., 2015). Formally, a confusion set can be defined as $f : (C \times C) \rightarrow \{\texttt{true}, \texttt{false}\}$. If $f(c_1, c_2) = \texttt{true}$, then c_1 and c_2 are considered easily confused. As pointed out by Liu et al. (2010), a faithful correction generally has phonetic and glyphic connections with the original characters. A typical confusion set might include characters with the same or similar pronunciation and similar shapes.

Input	记得戴眼睛***(jīng)***。 Remember to wear ***eyes***.
Reference	记得戴眼镜**(jìng)**。 Remember to wear **glasses**.
Input	从商场的人***(rén)***口进去。 Enter through the mall's ***population***.
Reference	从商场的入**(rù)**口进去。 Enter through the mall's **entrance**.

Table 1: Corrections marked in "**Blue**" are correct or suggested by the reference, while those in "***Red***" are incorrect. "睛"(jīng, eyes) and "镜"(jìng, glasses) are a pair of characters that are similar in phonetics, and "人"(human) and "入"(enter) are similar in glyph.

However, building an effective confusion set is challenging. If the threshold for determining whether two characters are confused is too strict, some similar characters may be missed. Conversely, if the threshold is too loose, the confusion set may contain too many characters, leading to over-correction.

Researchers have proposed various methods to address this issue. Some methods expand or improve confusion sets. For example, Huang et al. (2023) develop a module that distinguishes a character's errors as either phonological or visual, and then selects the correction from the corresponding confusion set. Other approaches leverage deep learning to internalize the confusion set rather than using it directly. Liu et al. (2021) employ a data

[1] Our anonymized code is available at https://anonymous.4open.science/r/simple-DISC.

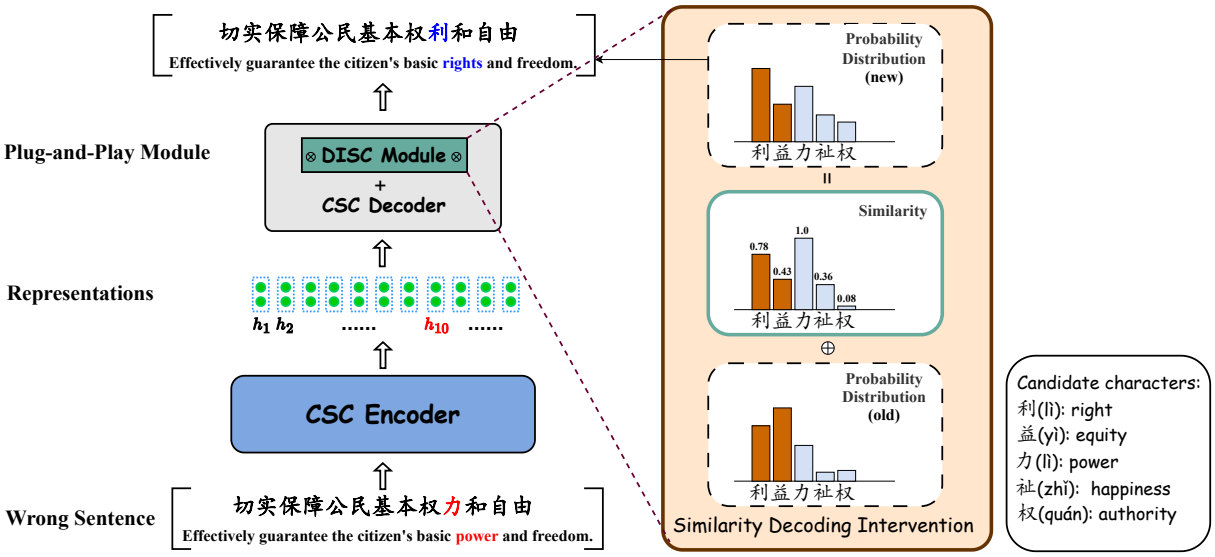


Figure 1: Overview of DISC. It intervenes the CSC decoder with the similarity between the potential error character and its candidate words. The DISC module intervenes in the probability distribution results of the CSC model with specific similarity, favoring the selection of more similar confusing characters.

augmentation method that replaces `[MASK]` with characters from a confusion set during model pre-training. Cheng et al. (2020) uses confusion sets to construct a graph convolutional network, modeling the connections between characters. Though these methods have achieved good results, they often lack intuitiveness and portability, and remain limited by the quality of the confusion set.

To overcome the binary classification limitations of confusion sets, we propose a plug-and-play and efficient DISC (**D**ecoding **I**ntervention with **S**imilarity of **C**haracters) module. This module introduces the similarity of characters into the decoding process, rather than relying on binary classification. As shown in Figure 1, DISC first calculates the similarities between the potentially erroneous characters and all possible characters in the vocabulary, and then intervenes in the probability distribution calculation process of the model decoding. Our module is compatible with almost all the current mainstream CSC models, such as SoftMasked-BERT (Zhang et al., 2020), ReaLiSe (Xu et al., 2021), SCOPE (Li et al., 2022), and ReLM (Liu et al., 2024), among others.

Experiments and analyses on multiple public benchmarks prove that our DISC module can significantly enhance the error correction performance of CSC models. This improvement does not require additional training costs and only slightly affects the decoding efficiency of the model.

The main contributions of this paper are summarized as follows:

- We design a plug-and-play DISC module that introduces external linguistic knowledge at the decoding end, compatible with almost all currently mainstream CSC models.
- We design simple and efficient strategies for calculating phonetic and glyph similarities, which are training-free and do not affect decoding efficiency.
- Extensive experiments on public benchmarks have shown that our DISC module can significantly enhance the performance of CSC models, achieving results that are close to or even exceed the state-of-the-art (SOTA) levels.

2 The Basic CSC Model

Given an input sentence consisting of n characters, denoted as $\boldsymbol{x} = x_1 x_2 \cdots x_n$, the goal of a CSC model is to output a corresponding correct sentence, denoted as $\boldsymbol{y} = y_1 y_2 \cdots y_n$, in which all erroneous characters in $\boldsymbol{x}$ are replaced with the correct ones.

Presently, mainstream approaches treat CSC as a character-wise classification problem (Zhang et al., 2020; Liu et al., 2021; Xu et al., 2021), i.e., determining whether a current character should be kept the same or be replaced with a new character.

Encoding. Given $\boldsymbol{x}$, the encoder of the CSC model generates representations for each character:

$$\mathbf{h}_1 \cdots \mathbf{h}_n = \texttt{Encoder}(\boldsymbol{x}). \quad (1)$$

To leverage the power of pre-trained language models, a BERT-like encoder is usually employed.

Classification. For each character position, for instance $\mathbf{h}_i$, the CSC model employs MLP and softmax layers to obtain a probability distribution over the whole character vocabulary $\mathcal{V}$:

$$p(y \mid \boldsymbol{x}, i) = \texttt{softmax}(\,\texttt{MLP}(h_i)\,)[y]. \quad (2)$$

During the evaluation phase, the model selects the character with the highest probability, i.e., $y^* = \arg\max_{y \in \mathcal{V}} p(y \mid \boldsymbol{x}, i)$.

Training. The typical training procedure consists of 2–3 steps for the CSC task. First, automatically synthesize large-scale CSC training data by replacing some characters with others randomly, sometimes constrained by a given confusion set. Second, train the CSC model on the synthesized training data. Third, fine-tune the model on a small-scale in-domain training data, if the data is available.

3 Our Approach

In this paper, we propose a simple plug-and-play module to intervene in the classification (or prediction) process of any off-the-shelf CSC model. The basic idea is to adjust the probability distribution according to the similarity between a candidate character y and the original character x_i:

$$\texttt{Score}(\boldsymbol{x}, i, y) = p(y \mid \boldsymbol{x}, i) + \alpha \times \texttt{Sim}(x_i, y), \quad (3)$$

where $\texttt{Sim}(\cdot)$ gives the similarity between two characters, and α is a hyperparameter and we set $\alpha = 1.1$ for all datasets and basic models according to a few preliminary experiments. We use $\texttt{Score}(\cdot)$ to denote the replacement likelihood since the value is no longer a probability.

Our experiments show that by encouraging the model to prefer similar characters, our approach achieves a consistent and substantial performance boost on all CSC benchmark datasets.

We measure character similarity from two perspectives, i.e., phonetic and glyph:

$$\begin{aligned}\texttt{Sim}(c_1, c_2) = \beta \times \texttt{Sim}^{\texttt{P}}(c_1, c_2) \\ +(1-\beta) \times \texttt{Sim}^{\texttt{G}}(c_1, c_2),\end{aligned} \quad (4)$$

where β is an interpolation hyperparameter, our experiments in Section 6 demonstrate that the model achieves good and stable performance when it is set to 0.7.

3.1 Phonetic Similarity

Given two characters, we employ the pypinyin library to obtain the Pinyin sequences,[2] e.g., "忠" (zhong) and "仲" (zhong),[3] and then compute the phonetic similarity based on the edit distance over their Pinyin sequences:

$$\texttt{Sim}^{\texttt{P}}(c_1, c_2) = 1 - \frac{\texttt{LD}(\texttt{py}(c_1), \texttt{py}(c_2))}{\texttt{len}(\texttt{py}(c_1) + \texttt{py}(c_2))}, \quad (5)$$

where $\texttt{LD}(\cdot)$ gives the Levenshtein distance,[4] and $\texttt{len}(\cdot)$ gives the total length of the two sequences.

Handling polyphonic characters. Given two characters, we enumerate all possible Pinyin sequences of each character, and adopt the combination that leads to the highest similarity.

We have also tried more sophisticated strategies. For instance, we follow Yang et al. (2023) and give higher weights to certain phoneme (consonant or vowel) pairs, since they are more likely to cause spelling errors. However, our preliminary experiments show that our simple strategy in Eq. 5 works quite robustly.

3.2 Glyph Similarity

According to Liu et al. (2010), 83% of Chinese spelling errors are related to pronunciation, while 48% are with glyphs, indicating that a considerable proportion is related to both. Therefore, it is necessary to consider the glyph information when computing character similarity.

Pinyin sequences can largely encode the phonetics of Chinese characters. In contrast, it is much more complex to represent character glyphs. In this work, we compute and fuse glyph similarity from four aspects:

$$\texttt{Sim}^{\texttt{G}}(c_1, c_2) = \frac{\sum_{i=1}^{4} \texttt{Sim}_i^{\texttt{G}}(c_1, c_2)}{4}. \quad (6)$$

Four-corner code. The four-corner method is widely used in Chinese lexicography for indexing characters. Given a character, it gives four digits ranging from 0 to 9, corresponding to the shapes

[2] https://pypi.org/project/pypinyin

[3] We do not use the tone information, e.g., "忠" (zhōng) and "仲" (zhòng), which is not helpful for model performance according to our preliminary experiments. We suspect the reason is that Pinyin-based input methods do not require users to input the tones. Therefore, tones are not directly related to spelling errors.

[4] Levenshtein distance is a type of edit distance. We set the weights of the three types of operations, i.e., deletion, insertion and substitutions, as 1/1/2 respectively.

at the four corners of the character's glyph, respectively. For instance, the four-corner code is 5033 for "忠", and 2520 for "仲".

Then, we use the digit-wise matching rate between two codes as the similarity:

$$\mathtt{Sim}_1^{\mathtt{G}}(c_1, c_2) = \frac{\sum_{i=1}^{4} \mathbb{1}(\mathtt{FC}(c_1)[i] = \mathtt{FC}(c_2)[i])}{4}, \tag{7}$$

where $\mathtt{FC}(\cdot)$ gives the four-digit code, and $\mathbb{1}$ is the indicator function.

Structure-aware four-corner code. One important feature of Chinese characters is that a complex character can usually be decomposed into simpler parts, and each part corresponds to a simpler character or a radical. Most radicals are semantically equivalent to some character, e.g., "亻" to "人".

Such structural decomposition directly reveals how characters are visually similar to each other. This is an effective way for Chinese learners to understand and memorize thousands of Chinese characters. Motivated by this observation, we design a structure-aware four-corner code for each character. For example,

"忠": C5000C3300 ("中": 5000; "心": 3300)
"仲": B8000B5000 ("人": 8000; "中": 5000)

where "C" leading a four-coner code means up-down structure, and "B" means left-right structure.

Then we compute the similarity based on the Levenshtein distance as follows:

$$\mathtt{Sim}_2^{\mathtt{G}}(c_1, c_2) = 1 - \frac{\mathtt{LD}(\mathtt{SFC}(c_1), \mathtt{SFC}(c_2))}{\mathtt{len}(\mathtt{SFC}(c_1) + \mathtt{SFC}(c_2))}, \tag{8}$$

where $\mathtt{SFC}(\cdot)$ gives the structure-aware code of a character.

Stroke sequences. Four-corner codes focus on the shapes of the four corners. Some very similar characters may obtain quite different codes, e.g., "木" (4090) vs. "本" (5023). To address this issue, we utilize stroke sequence information, which encodes how a character is handwritten stroke by stroke. For example,

"木": 一 丨 ノ 丶 (4 strokes)
"本": 一 丨 ノ 丶 一(5 strokes)

Then we compute two similarity metrics from two complementary viewpoints. The first metric is based on Levenshtein distance:

$$\mathtt{Sim}_3^{\mathtt{G}}(c_1, c_2) = 1 - \frac{\mathtt{LD}(\mathtt{SS}(c_1), \mathtt{SS}(c_2))}{\mathtt{len}(\mathtt{SS}(c_1) + \mathtt{SS}(c_2))}, \tag{9}$$

where $\mathtt{SS}(\cdot)$ gives the stroke sequence of a character.

The second metric considers the longest common subsequence, i.e., $\mathtt{LCS}(\cdot)$:

$$\mathtt{Sim}_4^{\mathtt{G}}(c_1, c_2) = \frac{\mathtt{LCS}(\mathtt{SS}(c_1), \mathtt{SS}(c_2))}{\max(\mathtt{len}(\mathtt{SS}(c_1)), \mathtt{len}(\mathtt{SS}(c_2)))}. \tag{10}$$

According to Eq. 4, and supposing $\beta = 0.7$, we get the similarity between "忠" and "仲" being:

$$0.7 \times 1 + 0.3 \times \frac{0 + 0.56 + 0.57 + 0.5}{4} = 0.82.$$

4 Experimental Setup

4.1 Datasets

Following the conventions of previous work, we employ the test sets of the SIGHAN 13/14/15 datasets (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) as our evaluation benchmarks.

However, many previous studies have pointed out that the SIGHAN datasets may not represent real-world CSC tasks, as they are derived from Chinese learner texts and lack diversity in terms of domains. To address this limitation, we also conduct experiments on the ECSpell (Lv et al., 2023) and LEMON (Wu et al., 2023) datasets, which are derived from Chinese native-speaker (CNS) texts and encompass a wide range of domains. It is worth noting that LEMON does not have a dedicated training set, making it an excellent test set for evaluating a model's generalization ability.

The details of these datasets are in Appendix B.

4.2 Baseline Models

We select three representative models as our baselines: **ReaLiSe**, **SCOPE**, and **ReLM**.

The **ReaLiSe** model (Xu et al., 2021) employs multi-modal technology to capture semantic, phonetic, and glyph information. The **SCOPE** model (Li et al., 2022) is one of the SOTA models for CSC, which enhances model correction performance by introducing a character pronunciation prediction task. The **ReLM** model (Liu et al., 2024) breaks away from the traditional CSC modeling approach by treating CSC as a non-autoregressive paraphrasing task, standing out as a new SOTA model.

Additionally, we include some of the latest work (Cheng et al., 2020; Liu et al., 2021; Huang et al., 2023) for performance comparison.

Dataset	Model	Detection-level			Correction-level			FPR
		P	R	F_1	P	R	F_1	
SIGHAN13	SpellGCN (Cheng et al., 2020)	80.1	74.4	77.2	78.3	72.7	75.4	–
	DR-CSC (Huang et al., 2023)†	88.5	83.7	86.0	87.7	83.0	85.3	–
	ReaLiSe (Xu et al., 2021)†	88.6	82.5	85.4	87.2	81.2	84.1	10.3
	+ DISC	88.9	82.2	85.4	87.6	81.1	84.2↑	10.3
	SCOPE (Li et al., 2022)†	87.2	82.7	84.9	86.3	81.8	84.0	10.3
	+ DISC	88.0	83.2	85.5↑	87.3	82.5	84.8↑	10.3
	ReLM (Liu et al., 2024)†	86.4	83.7	85.0	85.0	82.3	83.7	10.8
	+ DISC	**89.7**	**84.5**	**87.0**↑	**88.4**	**83.3**	**85.8**↑	**7.6**↓
SIGHAN14	SpellGCN (Cheng et al., 2020)	65.1	69.5	67.2	63.1	67.2	65.3	–
	DR-CSC (Huang et al., 2023)	70.2	73.3	71.7	69.3	72.3	70.7	–
	ReaLiSe (Xu et al., 2021)	67.8	71.5	69.6	66.3	70.0	68.1	14.9
	+ DISC	69.2	71.2	70.1↑	68.2	70.2	69.2↑	13.7↓
	SCOPE (Li et al., 2022)	67.6	71.9	69.7	66.7	70.9	68.8	14.9
	+ DISC	69.1	71.5	70.3↑	68.6	71.0	69.8↑	13.5↓
	ReLM (Liu et al., 2024)	66.1	72.9	69.3	64.3	70.9	67.4	15.5
	+ DISC	**70.4**	**73.3**	**71.8**↑	**69.5**	**72.3**	**70.8**↑	**12.7**↓
SIGHAN15	SpellGCN (Cheng et al., 2020)	74.8	80.7	77.7	72.1	77.7	75.9	–
	PLOME (Liu et al., 2021)	77.4	81.5	79.4	75.3	79.3	77.2	–
	DR-CSC (Huang et al., 2023)	**82.9**	84.8	**83.8**	**80.3**	82.3	81.3	–
	ReaLiSe (Xu et al., 2021)	77.3	81.3	79.3	75.9	79.9	77.8	12.0
	+ DISC	78.3	81.2	79.7↑	77.0	79.9	78.4↑	11.3↓
	SCOPE (Li et al., 2022)	81.0	84.8	82.9	78.7	82.4	80.5	11.1
	+ DISC	81.8	84.7	83.2↑	79.8	82.6	81.2↑	10.0↓
	ReLM (Liu et al., 2024)	78.3	**85.6**	81.8	76.8	**83.9**	80.2	12.7
	+ DISC	80.8	84.3	82.5↑	79.8	83.1	**81.4**↑	**9.5**↓

Table 2: Sentence-level performance on the SIGHAN13, SIGHAN14 and SIGHAN15 test sets. Precision (P), recall (R) and F_1 for detection and correction are reported (%). ReLM and SCOPE's results are obtained by reruning the official code released by Liu et al. (2024) and Li et al. (2022). Other baseline results are directly taken from their literature. Baselines marked with "†" mean that they apply post-processing on SIGHAN13, which removes all detected and corrected "地" and "得" from the model output before evaluation. "+ DISC" means adding DISC module in the decoder. α and β are assigned the values 1.1 and 0.7, respectively.

4.3 Evaluation Metrics

The CSC task comprises two subtasks: error detection and error correction. Following the previous work (Zhang et al., 2020), we report the precision (P), recall (R), and F_1 scores at the sentence level for both subtasks. Additionally, we also evaluate the models with the False Positive Rate (FPR) metric (Liu et al., 2024), which quantifies the CSC model's frequency of over-correction, i.e., incorrectly identifying correct sentences as erroneous.

4.4 Hyperparameters

Hyperparameters α and β denote the weights assigned to overall similarity and phonetic similarity, respectively. As detailed in Section 6 on grid search results, we set $\alpha = 1.1$ in Eq. 3 and β is set to 0.7 in Eq. 4 for all experiments.

5 Main Results

Results on SIGHANs. Table 2 illustrates the main results across SIGHAN benchmarks, demonstrating that the addition of the DISC module in the decoding processing leads to notable improvements across all the compared models, and reaching state-of-the-art performance. Specifically, ReaLiSe + DISC has increases of 0.1/1.1/0.6, SCOPE + DISC achieves lifts of 0.8/1.0/0.7, ReLM + DISC sees enhancements of 2.1/3.4/1.2 in correction-level F_1 (C-F_1) score on the SIGHAN13/14/15 test sets, respectively.

It is worth noting that ReaLiSe and SCOPE have incorporated phonetic or glyph information during training. However, our DISC module can still improve the performance of these models.

In addition to the consistent improvement in the

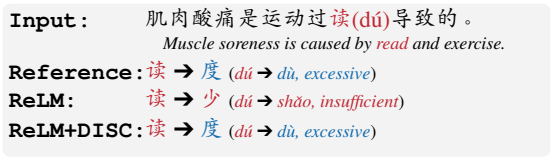
Domain	Model	Detection			Correction			FPR
		P	R	F_1	P	R	F_1	
ECSpell								
LAW	ReLM	93.7	98.8	96.2	93.7	98.8	96.2	6.5
	+ DISC	96.5	98.0	**97.3**	96.5	98.0	**97.3**	**2.9**
MED	ReLM	86.4	97.2	91.5	85.1	95.8	90.2	9.8
	+ DISC	92.9	97.7	**95.2**	91.6	96.3	**93.9**	**4.6**
ODW	ReLM	89.8	91.9	90.8	89.4	91.5	90.4	5.8
	+ DISC	91.5	91.5	**91.5**	91.1	91.1	**91.1**	**3.3**
LEMON								
GAM	ReLM	45.3	42.5	43.8	35.8	33.6	34.6	20.6
	+ DISC	62.2	34.9	**44.7**	56.1	31.5	**40.4**	**8.5**
CAR	ReLM	67.3	55.7	**60.9**	59.2	48.9	53.6	12.0
	+ DISC	76.0	48.2	59.0	72.3	45.9	**56.2**	**4.6**
NOV	ReLM	54.6	38.0	**44.8**	46.3	32.2	38.0	17.6
	+ DISC	70.2	31.9	43.9	65.2	29.6	**40.8**	**7.1**
ENC	ReLM	64.8	48.3	**55.4**	55.8	41.6	47.7	12.7
	+ DISC	76.3	41.5	53.8	72.2	39.3	**50.9**	**5.1**
NEW	ReLM	77.3	58.1	**66.3**	68.5	51.5	58.8	8.4
	+ DISC	85.6	51.2	64.1	80.4	48.1	**60.2**	**3.2**
COT	ReLM	82.8	70.7	**76.3**	73.5	62.8	67.7	4.9
	+ DISC	92.2	61.5	73.7	87.4	58.3	**69.9**	**1.1**
MEC	ReLM	74.3	49.6	59.4	67.3	44.9	53.9	5.8
	+ DISC	84.8	45.9	**59.6**	82.2	44.5	**57.7**	**2.2**
SIG	ReLM	65.3	63.5	64.4	59.4	57.8	58.6	16.3
	+ DISC	72.9	58.7	**65.0**	69.9	56.3	**62.4**	**8.9**

Table 3: Sentence-level performance of ReLM and ReLM + DISC on the test sets of ECSpell and LEMON.

F_1 metric, results demonstrate that the integration of the DISC module into CSC models leads to a significant reduction in FPR across almost all datasets. This implies that DISC can avoid some unnecessary corrections.

Results on Native Datasets. As ReLM has shown outstanding performance on the SIGHAN benchmarks, we continue to utilize it for experiments on the multi-domain datasets of ECSpell and LEMON to demonstrate the DISC module's domain adaptability.

Table 3 depicts that the incorporation of the DISC module on ReLM leads to substantial improvements of 1.1/3.7/0.7 C-F_1 score compared to unenhanced ReLM in the LAW, MED and ODW domain, respectively. It is worth noting that this improvement is achieved on the premise of surpassing a score of 90, which sufficiently illustrates the advancement of our DISC module. Table 3 also presents the performance of DISC on LEMON. After integrating the DISC module, the results of ReLM + DISC across all domains have achieved notable improvements, and the average C-F_1 have an increase of 3.2. This demonstrates that our DISC module yields stable and significant improvements in cross-domain CSC testing.



(a) Retrieve more similar candidate words

(b) Mitigate over-correction

Figure 2: Cases from the SIGHANs and ECSpell.

5.1 Case Study

We present two illustrative examples of DISC-augmented error correction in Figure 2. These examples explain why our DISC module can significantly improve model precision.

Figure 2(a) exemplifies how the DISC module retrieves a more plausible alteration resembling the original character. In this example, the ReLM model corrects the erroneous word "读"(dú) to "少"(shǎo). This correction is grammatically correct, but deviates from the original meaning of the sentence. From the perspective of phonetics, a more suitable correction should be "度"(dù), which shares the same pronunciation as the erroneous word. The DISC model can make this correction by considering the semantic and phonetic information of the sentence.

In Figure 2(b), the DISC alleviates over-correction. The CSC model mistakenly alters "肃"(sù) to "恕"(shù), yet the similarity intervention rectifies this error. Specifically, since the most similar to a character is the character itself, when a CSC model incorrectly tends to correct over preserve on a correct sentence, the DISC module can increase the score of the character itself compared to other correction options based on similarity, which sometimes avoids unnecessary corrections.

6 Discussion

We select the SIGHAN15 along with two domains from the LEMON database, ENC and MEC, to conduct further analysis.

Robustness of similarity hyperparameters. As illustrated in Figure 3, the incorporation of our

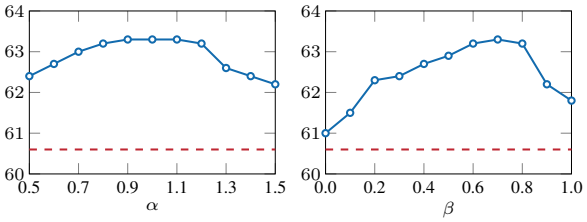


Figure 3: The average scores in ENC, MEC and SIGHAN15 with different values of α and β. The solid lines represent the results of ReLM + DISC, and the dashed lines represent the results of the original ReLM.

Model	ENC	MEC	SIG15	Avg
ReLM	47.7	53.9	80.2	60.6
+ DISC	50.9	**57.7**	**81.4**	**63.3**
+ Confusion set	47.1	56.0	78.0	60.4
+ Confusion set‡	41.5	48.7	80.7	57.0
+ DISC (phonetic)	49.1	56.1	80.1	61.8
+ DISC (glyph)	49.4	53.3	80.3	61.0
+ DISC (phonetic &)				
├$\texttt{Sim}^{\texttt{G}}_1$	50.5	56.8	81.4	62.9
├$\texttt{Sim}^{\texttt{G}}_2$	50.5	57.4	81.4	63.1
├$\texttt{Sim}^{\texttt{G}}_3$	51.3	57.5	81.2	**63.3**
└$\texttt{Sim}^{\texttt{G}}_4$	**51.6**	56.9	80.8	63.1

Table 4: Ablation results in two kinds of confusion sets and different components of DISC. "‡" represents the confusion set from Wang et al. (2019). "$\texttt{Sim}^{\texttt{G}}_i$" means using similarities of phonetic and the ith part of glyph.

DISC module into the ReLM model consistently improves its performance across a wide range of α and β values. The flat curve of α indicates that the performance of our DISC model is robust to the selection of α. In contrast, the value curve of β is relatively steep, showing a clear trend of rising first and then decreasing, reaching the highest value in the interval of $[0.6, 0.8]$. This phenomenon indicates that phonetic and glyph similarities are complementary, with phonetic similarity being relatively more important than glyph similarity.

Effectiveness of DISC module. We degrade the DISC module to a simple confusion set constraint decoding strategy. We investigate two confusion sets: one derived from our similarity computation strategy[5] and another pre-existing one provided by Wang et al. (2018). The results are shown in the second part of Table 4. From the results, we can see that both confusion sets fail to consistently improve the model's performance, indicating that the confusion set constraint decoding strategy is sensitive to the quality of the confusion set. The confusion set from Wang et al. (2018) yields improvements on SIGHAN15, yet we observe that it encompasses greater than 97% of the erroneous character pairs present in SIGHAN15. Conversely, it results in performance degradation on other test sets, highlighting the domain-specific limitations of such confusion sets.

[5]We treat a character pair as confused if their similarity score was greater than 0.5.

Model	Speed (ms/sent)	Slowdown
ReaLiSe	24.5	–
+ DISC	27.5	1.143×
SCOPE	138.6	–
+ DISC	143.4	1.035×
ReLM	12.7	–
+ DISC	12.8	1.010×

Table 5: The decoding time per sentence with a batch size of 1 on SIGHAN15. The results are the average time of three runs.

Effectiveness of components of the DISC module. We conduct an ablation study on the components of the DISC module. The results are shown in the third part of Table 4. The removal of either phonetic or glyph knowledge from the DISC module results in performance deterioration across various benchmarks. Notably, the absence of phonetic similarity has a lesser effect on SIGHAN15 but a stronger impact on LEMON. The results also show that the four components involved in calculating glyph similarity are independently effective. However, the exclusion of any three typically leads to a marginal decline in error correction performance, though exceptions do exist, such as ENC. This phenomenon verifies the necessity of multi-dimensional similarity measurement for comprehensive modeling of glyph similarity. Combining these often results in consistent improvements. Moreover, the fusion of phonetic and glyph similarities achieves the optimal error correction performance, affirming the necessity of integrating these two similarities.

Impact on decoding efficiency. We examine the influence of the DISC module on decoding speed, with the results shown in Table 5. Phonetic and glyph similarities can be pre-calculated and DISC only need to index them during decoding. Thus, the time taken to decode each sentence increased merely by 14.3%, 3.5%, and 1.0% for ReaLiSe, SCOPE, and ReLM, respectively. The minor slow-

down in decoding speed incurred by the DISC module is deemed acceptable considering the substantial enhancement it brings to the model's performance. Notably, SCOPE exhibits significantly slower decoding speeds compared to the other two models, which we speculate may be attributed to its iterative decoding approach.

7 Related Work

7.1 Model Architecture Shift

Most early works on CSC employed a three-step pipeline, i.e., 1) detecting potential erroneous characters, 2) constructing new sentences by replacing erroneous characters with new ones based on a confusion set; and 3) evaluating the probability of the constructed sentences based on an n-gram language model and choose the one with the highest probability (Yeh et al., 2013; Yu and Li, 2014; Huang et al., 2014; Xie et al., 2015).

In the current deep-learning era, especially with the prevalence of PLMs, recent models directly perform character-level replacement via classification, as introduced in Section 2. There also exist some works that employ a two-step pipeline architecture, which first detects potentially erroneous characters and then replaces them at the detected positions (Zhang et al., 2020; Huang et al., 2023).

7.2 Utilizing Confusion Sets

At both training and inferencing phases. Cheng et al. (2020) construct two character graphs, one based on phonetic relatedness, and the other based on glyph relatedness, and employ GCN to obtain new character representations as extra inputs. Huang et al. (2023) use two confusion sets, one encoding phonetic relatedness, and the other encoding glyph relatedness. Given a potential spelling error, they use a classification module to judge which confusion set the error belongs to, with an extra training loss. During the test phase, the model can only consider characters from the corresponding confusion set according to the classification result.

At only the inferencing phase. Wang et al. (2019); Bao et al. (2020) use the confusion set as constraints upon the search space, i.e., allowing the model to only consider characters in the confusion set. Please note that this is a decoding intervention technique as well, and is closely related to our work.

7.3 Utilizing Phonetic and Glyph Information

Besides the use of confusion sets, there exist some works that directly utilize phonetic and glyph information to enhance CSC models. Liu et al. (2021); Li et al. (2022) add an extra task of predicting the phonetic of each input character. Xu et al. (2021) use GRU to encode Pinyin, and use CNN to encode glyphs (font pictures) for each input character, as extra character representations.

7.4 Decoding Intervention

A typical decoding intervention approach is to use a language model to intervene in the decoding process of a sequence-to-sequence model. This idea has already been studied in many NLP tasks, including machine translation (Gülçehre et al., 2015), automatic speech recognition (Kannan et al., 2018; Zhao et al., 2019), and grammatical error correction (Zhou et al., 2023).

Besides using confusion sets for reducing search space, as discussed above, there exists an interesting work on the cross-domain CSC scenario. Lv et al. (2023) employ a word dictionary in the target domain to assist the decoding process.

8 Conclusions

We propose a streamlined, plug-and-play decoding intervention strategy that enhances CSC models by leveraging calculated phonetic and glyph similarities through a tailored algorithm. Unlike methods that alter model training, our training-free strategy only modifies the decoding process, making it adaptable to almost all mainstream CSC models. Experiments on multiple CSC benchmarks demonstrate that our method significantly enhances the performance of baseline models, and even surpasses the current SOTA CSC models. Furthermore, experimental analyses demonstrate that our DISC module aids the model in accurately identifying candidate characters which are more similar, effectively mitigating the issue of over-correction. Our research has transcended the limitations of traditional confusion set decoding intervention, proving that specific measures and combinations of phonetic and glyph similarities are necessary.

Limitations

We believe that our work can be further improved from two aspects. First, our experiments focus on the CSC datasets, while our approach can apply to other languages such as Japanese and Korean.

Second, as a general-use technique, our proposed approach for determining character similarity may not be optimal for CSC in specific domains or scenarios. In that case, we may need to consider more factors besides phonetic and glyph information to compute character similarity.

References

Zuyi Bao, Chen Li, and Rui Wang. 2020. Chunk-based Chinese Spelling Check with Global Optimization. In *Findings of EMNLP*, pages 2031–2040, Online.

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 ACL*, pages 871–881, Online.

Ming Dong, Yujing Chen, Miao Zhang, Hao Sun, and Tingting He. 2024. Rich Semantic Knowledge Enhanced Large Language Models for Few-shot Chinese Spell Checking. *ArXiv preprint*.

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 COLING*, pages 358–366, Beijing, China.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *ArXiv preprint*.

Haojing Huang, Jingheng Ye, Qingyu Zhou, Yinghui Li, Yangning Li, Feng Zhou, and Hai-Tao Zheng. 2023. A Frustratingly Easy Plug-and-Play Detection-and-Reasoning Module for Chinese Spelling Check. In *Findings of EMNLP*, pages 11514–11525, Singapore.

Qiang Huang, Peijie Huang, Xinrui Zhang, Weijian Xie, Kaiduo Hong, Bingzhou Chen, and Lei Huang. 2014. Chinese Spelling Check System Based on Tri-gram Model. In *Proceedings of CIPS-SIGHAN*, pages 173–178, Wuhan, China.

Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N. Sainath, Zhifeng Chen, and Rohit Prabhavalkar. 2018. An Analysis of Incorporating an External Language Model into a Sequence-to-Sequence Model. In *Proceedings of ICASSP*, pages 5824–5828, Calgary, Canada.

Jiahao Li, Quan Wang, Zhendong Mao, Junbo Guo, Yanyan Yang, and Yongdong Zhang. 2022. Improving Chinese Spelling Check by Character Pronunciation Prediction: The Effects of Adaptivity and Granularity. In *Proceedings of EMNLP*, pages 4275–4286, Abu Dhabi, United Arab Emirates.

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 *Proceedings of COLING*, pages 739–747, Beijing, China.

Linfeng Liu, Hongqiu Wu, and Hai Zhao. 2024. Chinese Spelling Correction as Rephrasing Language Model. In *Proceedings of AAAI*, pages 18662–18670, Vancouver, Canada.

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 ACL-IJCNLP*, pages 2991–3000, Online.

Qi Lv, Ziqiang Cao, Lei Geng, Chunhui Ai, Xu Yan, and Guohong Fu. 2023. General and Domain-adaptive Chinese Spelling Check with Error-consistent Pretraining. *TALLIP*, pages 1–18.

Bruno Martins and Mário J. Silva. 2004. Spelling Correction for Search Engine Queries. In *Proceedings of EsTAL*, pages 372–383, Alicante, Spain.

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 SIGHAN*, pages 32–37, Beijing, China.

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 EMNLP*, pages 2517–2527, Brussels, Belgium.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided Pointer Networks for Chinese Spelling Check. In *Proceedings of ACL*, pages 5780–5785, Florence, Italy.

Hongqiu Wu, Shaohua Zhang, Yuchen Zhang, and Hai Zhao. 2023. Rethinking Masked Language Modeling for Chinese Spelling Correction. In *Proceedings of ACL*, pages 10743–10756, Toronto, Canada.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese Spelling Check Evaluation at SIGHAN Bake-off 2013. In *Proceedings of SIGHAN*, pages 35–42, Nagoya, Japan.

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 SIGHAN*, pages 128–136, Beijing, China.

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 ACL-IJCNLP*, pages 716–728, Online.

Liner Yang, Xin Liu, Tianxin Liao, Zhenghao Liu, Mengyan Wang, Xuezhi Fang, and Erhong Yang. 2023. Is Chinese Spelling Check ready? Understanding the correction behavior in real-world scenarios. *AI Open*, pages 183–192.

Yifei Yang, Hongqiu Wu, and Hai Zhao. 2024. Attack Named Entity Recognition by Entity Boundary Interference. In *Proceedings of LREC-COLING*, pages 1734–1744, Torino, Italia.

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 SIGHAN*, pages 43–48, Nagoya, Japan.

Junjie Yu and Zhenghua Li. 2014. Chinese Spelling Error Detection and Correction Based on Language Model, Pronunciation, and Shape. In *Proceedings of CIPS-SIGHAN*, pages 220–223, Wuhan, China.

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 CIPS-SIGHAN*, pages 126–132, Wuhan, China.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling Error Correction with Soft-Masked BERT. In *Proceedings of ACL*, pages 882–890, Online.

Ding Zhao, Tara N. Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang. 2019. Shallow-Fusion End-to-End Contextual Biasing. In *Proceedings of INTERSPEECH*, pages 1418–1422, Graz, Austria.

Houquan Zhou, Yumeng Liu, Zhenghua Li, Min Zhang, Bo Zhang, Chen Li, Ji Zhang, and Fei Huang. 2023. Improving Seq2Seq Grammatical Error Correction via Decoding Interventions. In *Findings of EMNLP*, pages 7393–7405, Singapore.

A Implementation Details

We use the official implementation of ReaLiSe and directly utilize the checkpoint provided by its GitHub repository,[6] which initializes the semantic encoder with the weights of `chinese-roberta-wwm-ext`.[7] ReLM uses the official BERT weights `bert-base-chinese`,[8] and only offered the checkpoint after pre-training in 34 million monolingual sentences that are synthesized by confusion set. We fine-tune it on SIGHANs and ECSpell with a batch size of 128 and a learning rate of 3e-5, and the MFT strategy (Wu et al., 2023) is used during training. SCOPE utilizes the pre-trained weights from the `ChineseBERT-base`,[9] and we leverage their official implementation for fine-tuning.[10] Due to our decoding intervention strategy being deterministic, without any random factors, the experiments are conducted only once. All experiments are conducted on one Tesla V100S-PCIE-32GB GPU.

Training Set	#Sent	Avg. Length	#Errors
SIGHAN15	2,339	31.3	2,549
SIGHAN14	3,437	49.6	3,799
SIGHAN13	700	41.8	343
Wang271K	271,329	42.6	381,962
ECSpell_LAW	1,960	30.7	1,681
ECSpell_MED	3,000	50.2	2,260
ECSpell_ODW	1,720	41.2	1,578
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
ECSpell_LAW	500	29.7	390
ECSpell_MED	500	49.6	356
ECSpell_ODW	500	40.5	404
LEMON	22,252	35.4	12,055

Table 6: Statistics of the datasets, including the number of sentences, the average length of sentences, and the number of errors.

B Details of datasets

SIGHANs. Following the setup of previous work, we employ SIGHAN 13/14/15 datasets (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) as our training sets, in conjunction with Wang271K (Wang et al., 2018), which consists of 271K synthetically generated instances. We employ the test sets of SIGHAN13/14/15 for evaluation.

ECSpell. ECSpell (Lv et al., 2023) encompasses data from three domains: law, medical treatment, and official document writing. Unlike SIGHANs from Chinese learner texts, the sentences in ECSpell are derived from CNS texts.

LEMON. LEMON (Wu et al., 2023) also originates from CNS texts, containing over 22K instances spanning 7 domains. Given its lack of a dedicated training set, LEMON serves as a bench-

[6] https://github.com/DaDaMrX/ReaLiSe

[7] https://huggingface.co/hfl/chinese-roberta-wwm-ext

[8] https://huggingface.co/bert-base-chinese

[9] https://huggingface.co/ShannonAI/ChineseBERT-base

[10] https://github.com/jiahaozhenbang/SCOPE

mark for evaluating the domain adaptation capability of CSC models.

We conduct detailed statistics on the above datasets, and the results are presented in Table 6.