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

Anonymous ACL submission

Abstract

One key characteristic of the Chinese spelling check (CSC) task is that incorrect characters are usually similar to the correct ones in either phonetics or glyph. To accommodate this, previous works usually leverage confusion sets, which suffer from two problems, i.e., difficulty in determining which character pairs to include and lack of probabilities to distinguish items in the set. In this paper, we propose a light-weight plug-and-play DISC (i.e., decoding intervention with similarity of characters) module for CSC models. DISC measures phonetic and glyph similarities between characters and incorporates this similarity information only during the inference phase. 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 Introduction

Given an input sentence, Chinese spelling check (CSC) aims to detect incorrect characters and modify each into an correct character (Yu and Li, 2014; Xu et al., 2021). Table 1 gives two examples. Spelling errors degrade reading efficiency, and sometimes even lead to misunderstanding. The authority or attitude of the writer may be doubted if their document contains simple spelling errors. Moreover, spelling errors substantially hurt the performance of subsequent NLP models.

As is well known, spelling errors in Chinese texts have three major sources, i.e., 1) from keyboard typing with some input methods, 2) from image or document scanning with some optical character recognition (OCR) software, and 3) from speech-to-text translation with some automatic speech recognition (ASR) software. Nowadays, most Chinese users employ Pinyin-based input methods. Considering the three sources, we can see that the incorrect character in most cases is similar to the underlying correct one in phonetics or glyph, sometimes in both. This is a key characteristic of the CSC task.

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: Two CSC examples. "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.

Previous works employ confusion sets to leverage such similarities among characters (Yeh et al., 2013; Huang et al., 2014; Xie et al., 2015; Cheng et al., 2020; Huang et al., 2023). Formally, a confusion set is denoted as $\mathcal{C} = \{(c_i^1, c_i^2)\}_{i=1}^{M}$, where (c_i^1, c_i^2) represents a pair of characters and means that c_i^1 may be mistakenly replaced by c_i^2 in real texts.

As a representative work, Wang et al. (2018) construct a confusion set via two channels. First, they add noise into glyph images and apply OCR. Second, they apply ASR to parallel speech/text data. Their confusion set covers about 5K characters and consists of 19K character pairs that are likely to be confused with each other in written texts.

The most direct and popular use of confusion sets is to constrain the search space during the inference phase. The model can only consider character pairs in $\mathcal{C}$. More specifically, if $(c_1, c_2) \notin \mathcal{C}$, the model can never change c_2 into c_1. The justi-

fication for such *constrained decoding* is that the resulting sentence may deviate from the meaning of the input sentence (i.e., unfaithfulness), if the model replaces a character with a totally unrelated new one.

Despite their popularity and usefulness, confusion sets have two problems. First, it is difficult to set criteria to decide the inclusion or exclusion of certain character pairs. This renders the construction of confusion sets highly empirical, sometimes requiring manual intervention. Second, there is no probability to distinguish which character pairs are more likely to be confused than others in $\mathcal{C}$.

As a replacement for confusion sets, we propose a lightweight plug-and-play DISC (**d**ecoding **i**ntervention with **s**imilarity of **c**haracters) module. DISC derives probability-based similarities among characters in both phonetics and glyph, and uses the probabilities to intervene in the decoding process. Similar to the confusion set, our DISC aims to enhance the model's precision. However, for datasets that lack or have no in-domain training data, DISC may result in under-corrections due to the model's conservative predictions, leading to unstable recall. To address this, we propose a copy-punishment solution to balance precision and recall.

It is worth noting that DISC is featured in compatibility. On the one hand, DISC is compatible with the ways to derive probabilities for representing character similarity. On the other hand, DISC 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).

Experiments and analyses on popular benchmark datasets, i.e., SIGHANs, ECSpell, and LEMON, demonstrate 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. We release our code at `https://anonymous.4open.science/r/simple-DISC`.

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:

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

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

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

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

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:

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

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.

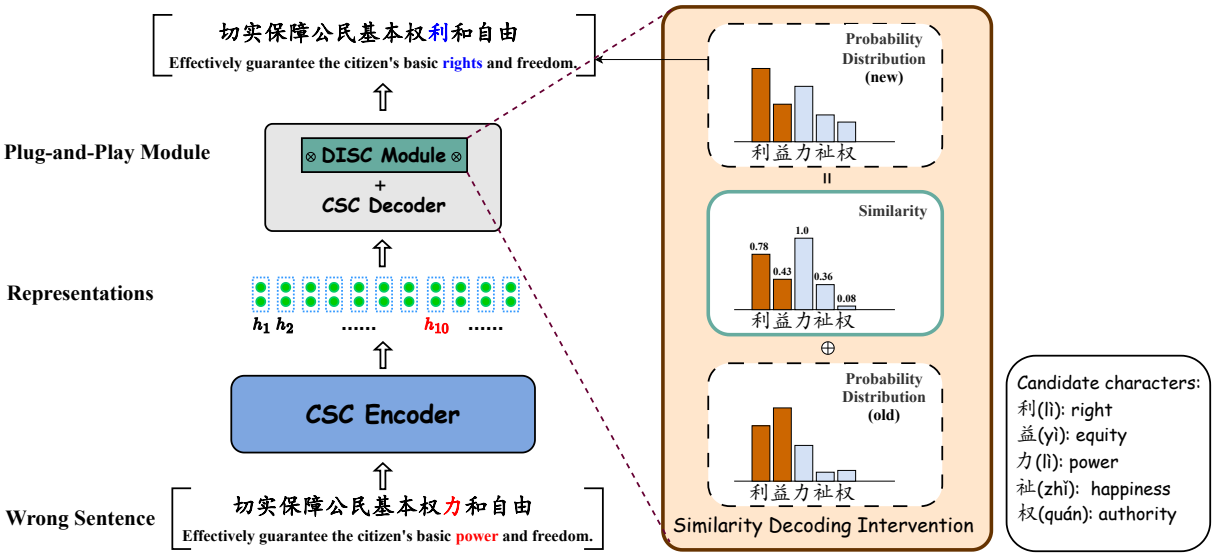

Figure 1: Overview of DISC. It intervenes in the CSC decoder with the similarity between the potential error character and its candidate characters. The DISC module intervenes in the probability distribution results of the CSC model based on specific similarity, favoring the selection of more similar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

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

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

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,[1] e.g., "忠" (zhong) and "仲" (zhong),[2] and then compute the phonetic similarity based on the edit distance over their Pinyin sequences:

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

where $\texttt{LD}(\cdot)$ gives the Levenshtein distance,[3] and $\texttt{len}(\cdot)$ gives the total length of the two sequences.

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

[2] 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.

[3] 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.

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. (2023b) 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:

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

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

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. 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. 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 BERT-style 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) treats 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; Huang et al., 2023) for performance comparison.

In the era of LLMs, researchers have begun using LLMs to explore the CSC field. We present the results of representative LLMs on certain benchmarks for comparison, including the top-performing GPT series in terms of overall capa-

Models	SIGHAN15				SIGHAN14				SIGHAN13			
	C-P$^{\uparrow}$	C-R$^{\uparrow}$	C-F$^{\uparrow}$	FPR$^{\downarrow}$	C-P$^{\uparrow}$	C-R$^{\uparrow}$	C-F$^{\uparrow}$	FPR$^{\downarrow}$	C-P$^{\uparrow}$	C-R$^{\uparrow}$	C-F$^{\uparrow}$	FPR$^{\downarrow}$
Previous SOTAs												
SpellGCN	72.1	77.7	75.9	–	63.1	67.2	65.3	–	78.3	72.7	75.4	–
ReaLiSe	75.9	79.9	77.8	12.0	66.3	70.0	68.1	14.9	87.2	81.2	84.1	10.3
SCOPE†	78.7	83.5	81.0	11.3	67.1	71.2	69.5	14.8	86.5	82.1	84.2	17.2
SCOPE + DR-CSC	**80.3**	82.3	81.3	–	69.3	72.3	70.7	–	87.7	83.0	85.3	–
ReLM†	76.8	**83.9**	80.2	12.7	63.7	72.3	67.7	17.5	85.0	82.3	83.7	10.8
LLMs Results												
GPT3.5	32.7	38.4	35.3	33.8	39.7	22.1	28.4	14.6	57.1	27.1	36.7	13.8
GPT4	36.5	49.2	41.9	40.8	32.8	45.0	38.0	43.5	47.3	45.7	46.5	44.8
Ours												
ReaLiSe + DISC	77.0	79.9	78.4$^{\uparrow}$	11.3$^{\downarrow}$	68.2	70.2	69.2$^{\uparrow}$	**13.7**$^{\downarrow}$	87.6	81.1	84.2$^{\uparrow}$	10.3
SCOPE + DISC	80.2	83.4	**81.8**$^{\uparrow}$	10.0$^{\downarrow}$	**69.3**	72.5	70.9$^{\uparrow}$	**13.7**$^{\downarrow}$	88.0	83.0	85.4$^{\uparrow}$	17.2
ReLM + DISC	79.8	83.1	81.4$^{\uparrow}$	**9.5**$^{\downarrow}$	68.6	**73.7**	**71.0**$^{\uparrow}$	14.3$^{\downarrow}$	**88.4**	**83.3**	**85.8**$^{\uparrow}$	**7.6**$^{\downarrow}$

Table 2: Sentence-level performance on the SIGHAN13, SIGHAN14 and SIGHAN15 test sets. Precision (P), recall (R) and F_1 for correction are reported (%). Results marked with "†" are obtained by reruning the official code released by Li et al. (2022) and Liu et al. (2024). Other baseline results are directly taken from their literature. Apart from SpellGCN, all models 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.

bility: GPT3.5 and GPT4, as well as some results on open-source LLMs in the Chinese NLP community from previous work, such as finetuned Baichuan2 (Yang et al., 2023a).

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 $\beta = 0.7$ 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 process leads to notable improvements across all the compared models, reaching state-of-the-art performance. Specifically, ReaLiSe + DISC has increases of 0.1/1.1/0.6, SCOPE + DISC achieves lifts of 1.2/1.4/0.8, ReLM + DISC sees enhancements of 2.1/3.3/1.2 in correction-level F_1 (C-F) 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 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 into ReLM leads to substantial improvements of 1.1/3.7/0.7 C-F score compared to unenhanced ReLM in the LAW, MED and ODW domains, respectively. Table 3 also presents the performance of DISC on LEMON. After integrating the DISC module, the results of ReLM + DISC achieve notable improvements across all domains, and the average C-F has an increase of 3.2. This

Domain	Model	Correction			FPR
		P	R	F_1	
ECSpell					
LAW	GPT3.5	48.5	43.1	45.6	9.4
	GPT4	62.0	62.0	62.0	7.3
	Baichuan2-7B⋆	85.1	87.1	86.0	–
	ReLM	93.7	**98.8**	96.2	6.5
	+ DISC	**96.5**	98.0	**97.3**	**2.9**
MED	GPT3.5	36.5	42.0	39.1	20.1
	GPT4	45.1	57.5	50.6	24.8
	Baichuan2-7B⋆	72.6	73.9	73.2	–
	ReLM	85.1	95.8	90.2	9.8
	+ DISC	**91.6**	**96.3**	**93.9**	**4.6**
ODW	GPT3.5	57.3	52.3	54.7	6.3
	GPT4	71.7	67.6	69.5	**1.7**
	Baichuan2-7B⋆	86.1	79.3	82.6	–
	ReLM	89.4	**91.5**	90.4	5.8
	+ DISC	**91.1**	91.1	**91.1**	3.3
LEMON					
GAM	ReLM	35.8	**33.6**	34.6	20.6
	+ DISC	**56.1**	31.5	**40.4**	**8.5**
CAR	ReLM	59.2	**48.9**	53.6	12.0
	+ DISC	**72.3**	45.9	**56.2**	**4.6**
NOV	ReLM	46.3	**32.2**	38.0	17.6
	+ DISC	**65.2**	29.6	**40.8**	**7.1**
ENC	ReLM	55.8	**41.6**	47.7	12.7
	+ DISC	**72.2**	39.3	**50.9**	**5.1**
NEW	ReLM	68.5	**51.5**	58.8	8.4
	+ DISC	**80.4**	48.1	**60.2**	**3.2**
COT	ReLM	73.5	**62.8**	67.7	4.9
	+ DISC	**87.4**	58.3	**69.9**	**1.1**
MEC	ReLM	67.3	**44.9**	53.9	5.8
	+ DISC	**82.2**	44.5	**57.7**	**2.2**
SIG	ReLM	59.4	**57.8**	58.6	16.3
	+ DISC	**69.9**	56.3	**62.4**	**8.9**

Table 3: Sentence-level performance of LLMs, ReLM, and ReLM + DISC on the test sets of ECSpell and LEMON. Results marked with "⋆" are from Liu et al. (2024).

demonstrates that our DISC module yields stable and significant improvements in cross-domain CSC testing.

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 makes this correction by leveraging semantic and phonetic information.

(a) Select the more similar word

(b) Mitigate over-correction

Figure 2: Cases from the SIGHANs and LEMON.

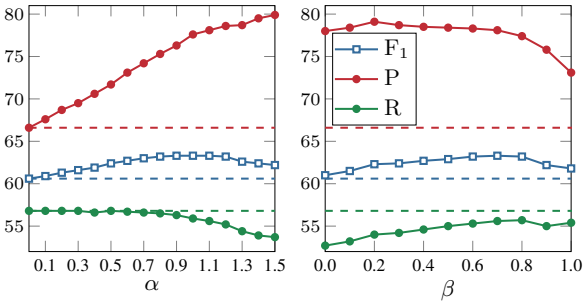

Figure 3: The average scores in ENC, MEC and SIGHAN15 with different values of α and β. The solid lines represent the results of ReLM + DISC, and the dashed lines represent the results of the original ReLM.

In Figure 2(b), the DISC alleviates over-correction. The CSC model mistakenly alters "空"(kōng) to "日"(rì), 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 model's precision steadily improves as α increases. This is because increased similarity intervention reduces

Domain	Edit Pairs	Seen Pairs	Prop.
SIGHANs			
SIGHAN15	703	698	99.29%
SIGHAN14	771	765	99.22%
SIGHAN13	1,224	1,206	98.53%
ECSpell			
LAW	390	321	82.31%
MED	356	265	74.44%
ODW	404	341	84.41%
LEMON			
GAM	164	100	60.98%
CAR	1,911	1,254	65.62%
NOV	3,415	2,045	59.88%
ENC	1,787	1,040	58.20%
NEW	3,260	2,293	70.34%
COT	486	309	63.58%
MEC	1,032	627	60.76%

Table 4: Proportion of seen edit pairs in the test sets of SIGHANs, ECSpell, and LEMON.

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 5: Ablation results in two kinds of confusion sets and different components of DISC. "‡" represents the confusion set from Wang et al. (2018). "$\texttt{Sim}^{\texttt{G}}_i$" means using similarities of phonetics and the ith part of glyph.

over-correction (Figure 2(a)), boosting precision. However, at the same time, DISC may revert predictions to the original character, as characters are most similar to themselves. This under-correction phenomenon caused by DISC sometimes leads to instability in recall.

For β, it shows the effect of the proportion of phonetic similarity in the total similarity on the model's correction performance. The F_1 score curve shows a clear trend of rising first and then decreasing, which indicates that phonetic and glyph similarities are complementary, with phonetic similarity being relatively more important than glyph similarity.

The balance between precision and recall. The primary purpose of using a confusion set is to narrow down the retrieval space, thereby improving precision. While a confusion set can enhance recall by enabling the model to make more reasonable edits, it may also reduce recall by discouraging the model from making edits, because the most similar character to the source character is itself.

We observe that this decrease in recall primarily occurs in the LEMON test set. The key distinction between these datasets is that LEMON contains less seen edit pairs in training data. As shown in Table 4, we calculate the proportion of edit pairs from each test set that appear in the training set.[4] Models are prone to copy the source character when the target edit pair is not available in the training data.

For this type of test set, we can use a simple copy punishment combined with DISC, which reduces the probability of copying the original character during inference, to mitigate the decrease in recall. Detailed experimental results can be found in the Appendix C.

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 5. From the results, we can see that both confusion sets fail to consistently improve performance, indicating the strategy's sensitivity to confusion set quality. The confusion set from Wang et al. (2018) improves SIGHAN15 by covering over 99% of its erroneous pairs but degrades performance on other test sets, highlighting the domain-specific limitations of such confusion sets.

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 5. Removing either phonetic or glyph knowledge from the DISC module leads to performance declines across 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, ex-

[4]For LEMON and ECSpell, we conduct statistics using the pairs in the confusion set which is used to generate 34 million monolingual sentences.

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

cluding any three typically causes a slight drop in performance, with exceptions like ENC. This phenomenon underscores the necessity of using multi-dimensional similarity measurements for a more 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.

7 Related Work

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).

Utilizing confusion sets. These works fall into three categories. (1) *At only the inference 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.

(2) *For data synthesis.* Liu et al. (2021) use a confusion set $\mathcal{C}$ to synthesize data for training CSC models. For a given correct sentence, they randomly select a character (e.g., c_i), and replace it with an incorrect character (e.g., c'). Only characters in the confusion set, i.e., $(c_i, c') \in \mathcal{C}$, are considered.

(3) *At both training and inference 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.

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.

Decoding intervention. Gou and Chen (2021) extract features such as probability and rank of the original character and the top 1 candidate character, and use SVM to determine whether the modification should be retained. Yin et al. (2024) first retrieve similar segments from the training set. Then, they intervene in the decoding process based on the segment (n-gram) similarity between the retrieved segments and the input. Lv et al. (2023) employ a word dictionary in the target domain to assist the decoding process.

8 Conclusions

We propose a plug-and-play decoding intervention strategy that enhances CSC models by utilizing 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 improves baselines, and even surpasses the current SOTA models. Furthermore, experimental analyses demonstrate that our DISC module helps the model better identify similar candidate characters, effectively reducing 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.

Wei Gou and Zheng Chen. 2021. Think twice: A post-processing approach for the chinese spelling error correction. *Applied Sciences*, 11(13).

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.

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.

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.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023a. Baichuan 2: Open large-scale language models.

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

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.

Xunjian Yin, Xinyu Hu, Jin Jiang, and Xiaojun Wan. 2024. Error-Robust Retrieval for Chinese Spelling Check. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6257–6267, Torino, Italia. ELRA and ICCL.

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.

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] We did not attempt DR-CSC + DISC because they have not fully open-sourced their work. 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.

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.

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.

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 benchmark 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.

C Copy Punishment

For datasets like LEMON that lack in-domain training data, we discover a simple recall-boosting solution: reducing the probability of selecting the original character during inference. Specifically, after incorporating the DISC module, we additionally lower the prediction probability of the original character by 0.1 to reduce the model's tendency to select the original character during inference, thereby improving the model's recall rate. The experimental results can be found in Table 7.

[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

Domain	Model	Correction			FPR
		P	R	F_1	
LEMON					
GAM	ReLM	35.8	33.6	34.6	20.6
	+ DISC	**56.1**	31.5	40.4	**8.5**
	+ DISC*	52.4	**37.0**	**43.4**	11.3
CAR	ReLM	59.2	**48.9**	53.6	12.0
	+ DISC	**72.3**	45.9	**56.2**	**4.6**
	+ DISC*	68.0	47.5	55.9	6.2
NOV	ReLM	46.3	**32.2**	38.0	17.6
	+ DISC	**65.2**	29.6	**40.8**	**7.1**
	+ DISC*	57.8	31.2	40.6	10.2
ENC	ReLM	55.8	41.6	47.7	12.7
	+ DISC	**72.2**	39.3	50.9	**5.1**
	+ DISC*	66.9	**41.7**	**51.4**	7.1
NEW	ReLM	68.5	**51.5**	58.8	8.4
	+ DISC	**80.4**	48.1	60.2	**3.2**
	+ DISC*	76.5	49.7	**60.3**	4.3
COT	ReLM	73.5	**62.8**	67.7	4.9
	+ DISC	**87.4**	58.3	**69.9**	**1.1**
	+ DISC*	80.8	61.0	69.5	1.8
MEC	ReLM	67.3	44.9	53.9	5.8
	+ DISC	**82.2**	44.5	**57.7**	**2.2**
	+ DISC*	76.3	**45.0**	56.6	3.2

Table 7: Sentence-level performance of LLMs, ReLM, and ReLM + DISC on the test sets of ECSpell and LEMON. Results marked with "*" indicate the use of copy-punishment solution.

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

D Impact on decoding efficiency

We examine the influence of the DISC module on decoding speed, with the results shown in Table 8. Phonetic and glyph similarities can be precalculated 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 slowdown 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.

System and User Prompts for LLMs

System Prompt:
你是一个优秀的中文拼写纠错模型，中文拼写纠错模型即更正用户输入句子中的拼写错误。
User Prompt:
你需要识别并纠正用户输入的句子中可能的错别字并输出正确的句子，纠正时必须保证改动前后句子等长，在纠正错别字的同时尽可能减少对原句子的改动(不添加额外标点符号，不添加额外的字，不删除多余的字)。只输出没有错别字的句子，不要添加任何其他解释或说明。如果句子没有错别字，就直接输出和输入相同的句子。

Figure 4: Prompt templates used in GPT3.5 and GPT4.

E Prompt Examples

In this work, we use the prompt-based method to activate the CSC ability of the `GPT3.5` and `GPT4`. The prompt used for the baselines are shown in Figure 4.

F More Results

In addition to the correction-level performance, which is our primary focus, we also present the detection-level experimental results of the CSC models, as shown in Table 9. SCOPE + DR-CSC performs well at the detection level, primarily because they incorporate an additional detection network.

Since SIGHANs contains a lot of noise, we also conduct experiments on their revised versions (referred to as SIGHANs (rev.)) released by Yang et al. (2023b), which have undergone manual verification and error correction to ensure higher data quality. As shown in Table 10 and Table 11, our DISC module also achieves consistent performance improvements on SIGHANs (rev.).

Models	SIGHAN15			SIGHAN14			SIGHAN13		
	D-P↑	D-R↑	D-F↑	D-P↑	D-R↑	D-F↑	D-P↑	D-R↑	D-F↑
Previous SOTAs									
SpellGCN	74.8	80.7	77.7	65.1	69.5	67.2	80.1	74.4	77.2
ReaLiSe	77.3	81.3	79.3	67.8	71.5	69.6	88.6	82.5	85.4
SCOPE†	80.5	85.4	82.9	68.8	73.7	71.1	87.5	83.0	85.2
SCOPE + DR-CSC	**82.9**	84.8	**83.8**	**70.2**	73.3	71.7	88.5	83.7	86.0
ReLM†	78.3	**85.6**	81.8	65.7	74.5	69.8	86.4	83.7	85.0
LLMs Results									
GPT3.5	39.4	46.4	42.6	41.4	23.1	29.6	61.6	29.2	39.7
GPT4	42.7	57.5	49.0	38.1	52.3	44.1	53.4	51.6	52.5
Ours									
ReaLiSe + DISC	78.3	81.2	79.7↑	69.2	71.2	70.1↑	88.9	82.2	85.4
SCOPE + DISC	81.7	84.8	83.2↑	**70.2**	73.5	71.8↑	88.8	83.7	86.2↑
ReLM + DISC	80.8	84.3	82.5↑	69.7	**74.9**	**72.2**↑	**89.7**	**84.5**	**87.0**↑

Table 9: Sentence-level performance on the SIGHAN13, SIGHAN14 and SIGHAN15 test sets. Precision (P), recall (R) and F_1 for detection are reported (%). Results marked with "†" are obtained by reruning the official code released by Li et al. (2022) and Liu et al. (2024). Other baseline results are directly taken from their literature. Apart from SpellGCN, all models 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.

Models	SIGHAN15 (rev.)				SIGHAN14 (rev.)				SIGHAN13 (rev.)			
	C-P↑	C-R↑	C-F↑	FPR↓	C-P↑	C-R↑	C-F↑	FPR↓	C-P↑	C-R↑	C-F↑	FPR↓
Previous SOTAs												
BERT⋆	73.2	67.5	70.2	–	62.6	57.5	59.9	–	71.1	67.4	69.2	–
ReaLiSe⋆	74.4	69.6	71.9	–	63.6	59.0	61.2	–	71.9	68.0	69.9	–
Yang et al. (2023b)	77.0	67.6	72.0	–	66.0	57.1	61.3	–	73.2	67.1	70.0	–
ReLM	76.4	**73.5**	74.9	8.5	65.5	62.9	64.2	11.3	74.0	70.9	72.4	10.7
Ours												
ReLM + DISC	**79.0**	73.0	**75.9**	**6.4**	**69.3**	**63.4**	**66.2**	**8.9**	**75.4**	**71.3**	**73.3**	**9.4**

Table 10: Sentence-level performance on the revised SIGHAN13-15 test sets. Precision (P), recall (R) and F_1 for correction are reported (%). "*" means that the results of BERT and ReaLiSe in the table are directly copied from Yang et al. (2023b).

Models	SIGHAN15 (rev.)			SIGHAN14 (rev.)			SIGHAN13 (rev.)		
	D-P↑	D-R↑	D-F↑	D-P↑	D-R↑	D-F↑	D-P↑	D-R↑	D-F↑
Previous SOTAs									
BERT⋆	75.4	70.0	72.4	64.6	59.3	61.8	72.6	68.8	70.6
ReaLiSe⋆	75.8	70.9	73.2	65.6	60.8	63.1	74.9	70.7	72.7
Yang et al. (2023b)	77.7	68.3	72.7	67.2	58.1	62.3	74.4	68.3	71.2
ReLM	78.8	**75.7**	77.2	68.4	**65.7**	67.0	76.0	**72.8**	**74.4**
Ours									
ReLM + DISC	**80.6**	74.4	**77.4**	**71.2**	65.2	**68.1**	**76.4**	72.3	74.3

Table 11: Sentence-level performance on the revised SIGHAN13-15 test sets. Precision (P), recall (R) and F_1 for detection are reported (%). "*" means that the results of BERT and ReaLiSe in the table are directly copied from Yang et al. (2023b).