A Simple yet Effective Training-free Prompt-free Approach to Chinese Spelling Correction Based on Large Language Models

Anonymous ACL submission

Abstract

This work proposes a simple yet effective approach for leveraging large language models (LLMs) in Chinese spelling correction (CSC) task. Our approach consists of two components: a LLM and a minimal distortion model. At each decoding step, the LLM calculates the probabilities of the next token based on the preceding context. Then, the distortion model adjusts these probabilities to penalize the generation of tokens that deviate too far from the input. Different from the prior supervised fine-tuning and prompt-based approaches, our approach enables efficient CSC without requiring additional training or task-specific prompts. To address practical challenges, we propose a length reward strategy to mitigate the local optima problem during beam search decoding, and a faithfulness reward strategy to reduce over-corrections. Comprehensive experiments on five public datasets demonstrate that our approach significantly improves LLM performance, enabling them to compete with state-of-the-art domain-general CSC models.[1]

1 Introduction

Spelling errors are common in Chinese text because many Chinese characters have similar pronunciations or shapes. This similarity makes it difficult for both humans to type and for machines to recognize the characters correctly. These errors may cause misunderstandings, diminish the credibility, or degrade the performance of downstream applications (Si et al., 2023). Therefore, the research on Chinese Spelling Correction (CSC) has become urgently necessary and attracted increasing attention in recent years (Hong et al., 2019; Bao et al., 2020; Xu et al., 2021; Li et al., 2022; Wu et al., 2023; Dong et al., 2024, *inter alia*).

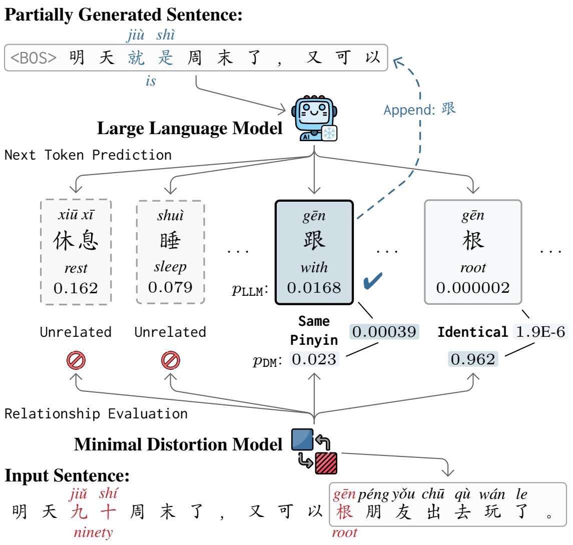


Figure 1: An illustration of our approach. The correct sentence should be "明天就是周末了，又可以跟朋友出去玩了。" (*Tomorrow is the weekend, allowing for going out to play with friends again.*).

Recently, researchers propose to leverage large language models (LLMs) to improve CSC performance. These approaches fall into two categories: *prompt-based* and *supervised fine-tuning*. The prompt-based approaches, which are widely used in the LLM era, feed CSC-related instructions and the input sentence into an LLM, and expect the LLM to output a corrected sentence. The experiment setting is called few-shot if a few CSC examples are included in the instructions, and zero-shot if no examples are provided. Li et al. (2023a) first investigate the prompt-based approach and conduct extensive experiments under different settings. Moreover, they propose different strategies for selecting proper examples. Dong et al. (2024) follow the work of Li et al. (2023a), and propose to enrich the prompt with additional information, such as pronunciation and glyph of characters. All their experiments show that the prompt-based approach leads to unsatisfactory CSC performance,

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

especially when compared to previous non-LLM based approaches.

The second class of approaches are based on supervised fine-tuning (SFT). The main difference between the prompt-based and the SFT-based approaches is the latter fine-tunes the LLM over the CSC training data. This SFT is performed one mini-batch at a time, with output corrected sentences as the training objective, in a teacher-forcing manner. Li et al. (2023a) explore the SFT-based approach under various settings and using different strategies. They find that the SFT-based approach achieve better performance than the prompt-based approach. However, the performance still lags behind previous non-LLM results by large margin.

In contrast to both the prompt-based and SFT-based approaches, we propose a simple prompt-free and training-free framework to leverage LLMs for the CSC task. As shown in Figure 1, our approach consists of two components: a LLM and a distortion model. At each decoding step, the LLM generates a token based on the current context. Then a minimal distortion model determines whether the generated token is deviated too far from the input characters. In practice, we find that the local optima problem of beam search decoding and over-correction hinder the performance of our approach. To address these issues, we propose two straightforward rewards, the length reward and faithfulness reward.

We conduct comprehensive experiments on five public datasets from various domains and genres, including more than 50,000 sentences. The results clearly show that our approach significantly improves the performance of LLMs in the CSC task. Our approach also demonstrates remarkable domain generalization capabilities, outperforming state-of-the-art domain-general CSC models trained on extensive synthetic CSC data (approximately 34 million pairs) on most datasets.

In summary, our contributions are as follows:

- We propose a simple yet effective framework to leverage LLMs for the CSC task, requiring neither additional training nor prompts.
- Two straightforward rewards, the length reward and faithfulness reward, are introduced to address the local optima problem and over-correction issue, respectively.
- Comprehensive experiments demonstrate that our approach significantly improves the performance of LLMs in the CSC task, showcasing remarkable domain generalization capabilities.

Type	**Example**		**Proportion**
`Identical`	机	(*jī*)	0.962
`Same Pinyin`	基	(*jī*)	0.023
`Similar Pinyin`	七	(*qī*)	0.008
`Similar Shape`	仉	(*zhǎng*)	0.004
`Unrelated`	能	(*néng*)	0.003

Table 1: Examples of the different distortion types of the corrected token “机” (*jī*). The distribution of the types is calculated from the development set.

2 Our Approach

Given an input sentence $\boldsymbol{x} = x_1, x_2, \cdots, x_n$, where x_i denotes a character, a CSC model outputs a sentence of the same length, denoted as $\boldsymbol{y} = y_1, y_2, \cdots, y_n$. The key to the CSC task is how to model the score of the input and output sentence pair, i.e., $\texttt{score}(\boldsymbol{x}, \boldsymbol{y})$.

Under a perspective of probabilistic modeling, the joint probability can be decomposed into two parts:

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y}) &= p(\boldsymbol{x} \mid \boldsymbol{y})\, p(\boldsymbol{y}) \\ &= p_{\texttt{DM}}(\boldsymbol{x} \mid \boldsymbol{y})\, p_{\texttt{LLM}}(\boldsymbol{y}) \end{aligned} \tag{1}$$

The first part corresponds to a distortion model, which captures the relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$. In other words, it interprets how spelling errors transform $\boldsymbol{y}$ to $\boldsymbol{x}$. Another important function of the distortion model is to make sure that $\boldsymbol{y}$ represents the same “meaning” as $\boldsymbol{x}$, i.e., faithfulness.

The second part corresponds to a large language model, which makes sure that $\boldsymbol{y}$ is fluent and correct from the language use perspective. In this work, we employ generative LLMs, including `Baichuan2`, `Qwen1.5`, and `InternLM2`.

Please note that our use of LLMs is **prompt-free**. We do not provide CSC-related instructions and examples as the prompt. More importantly, we do not give the input sentence to LLMs. We use LLMs as pure traditional language models for evaluating next-token probabilities.

2.1 A Minimal Distortion Model

Our distortion model adopts character-level factorization:

$$\log p_{\texttt{DM}}(\boldsymbol{x} \mid \boldsymbol{y}) = \sum_i \log p_{\texttt{DM}}(x_i \mid y_i) \tag{2}$$

To further simplify the model, we **do not** compute distortion probabilities for specific character pairs, i.e., (c_1, c_2). Instead, we first classify

(c_1, c_2) into one of five distortion types, denoted as $\texttt{type}(c_1, c_2)$. Then we use the probability of the type as the distortion probability of the character pair:

$$p_{\text{DM}}(c_1 \mid c_2) = p(\texttt{type}(c_1, c_2)) \tag{3}$$

Table 1 illustrates the distortion types. The proportions are obtained from small subsets of popular CSC training data, described later in §3.1. We directly employ the proportions as the distortion probabilities.

Please note that we claim our approach as **training-free**, since the LLMs are used in an off-the-shelf manner and the distortion model only relies on several frequency values, which can be easily counted from a small dataset.

Given (c_1, c_2), we implement a simple rule-based tool to decide the distortion type. Among the five types, "Similar Pinyin" and "Similar Shape" are more complex to handle. We give details in Appendix A, and release the tool, along with other code in this work.

2.2 Next-token Probabilities from LLM

Typically, the output vocabulary of a LLM contains both single- and multi-character tokens. In other words, given a sentence $\boldsymbol{y} = y_1 ... y_n$, there exists many ways to segment it into a sequence of tokens. We use $\boldsymbol{t} = t_1 ... t_m$ to denote a specific token-level segmentation of $\boldsymbol{y}$, i.e., a path for the LLM to generate the character sequence, where $t_j = c_1 \ldots c_k$ and $k \geqslant 1$. Then, the log probability of $\boldsymbol{y}$ can be decomposed as:

$$\log p_{\text{LLM}}(\boldsymbol{y}) = \sum_j \log p_{\text{LLM}}(t_j \mid \boldsymbol{t}_{<j}) \tag{4}$$

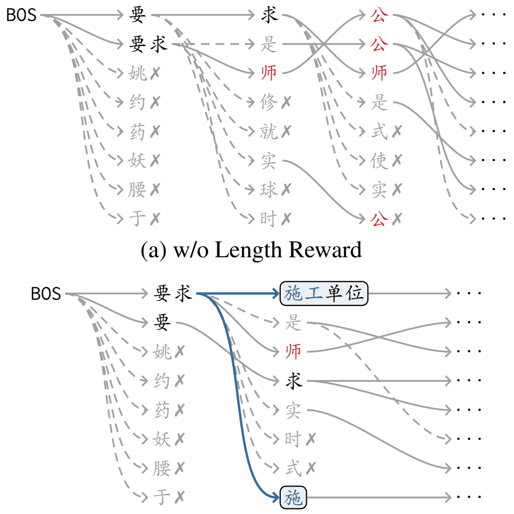
After combining the distortion model, the probability of a partial output sentence is:

$$\begin{aligned} \log p(\boldsymbol{x}, \boldsymbol{t}_{\leqslant j}) = \; & \log p(\boldsymbol{x}, \boldsymbol{t}_{<j}) \\ & + \log p_{\text{LLM}}(t_j \mid \boldsymbol{t}_{<j}) \\ & + \sum_{r=1}^{k} \log p_{\text{DM}}(c_r \mid x_{l+r}) \end{aligned} \tag{5}$$

where $k = \ell(t_j)$ and $l = \ell(\boldsymbol{t}_{<j})$ are the lengths of t_j and $\boldsymbol{t}_{<j}$, respectively.

2.3 Beam Search Decoding

During inference, the basic operation at step j is to select a token t_j and append it to the current partial sequence $\boldsymbol{t}_{<j}$. We follow the standard practice, and adopt beam search decoding, that only retains the top-K candidates at each decoding step for computational efficiency.



(a) w/o Length Reward

(b) w/ Length Reward

Figure 2: A real example of the decoding process for the input sentence "要求师公单位对..." (*Requesting the master unit to ...*). Here, "施工" (*shīgōng*, *construction*) is misspelled as "师公" (*shīgōng*). Without the length reward, the correct character "施" is fail to be select into the beam.

In particular, one technical detail is closely related with our length reward strategy and thus worthy of further discussion. As discussed above, most LLMs generate sentences at token-level and one token may contain either a single character or multiple characters. This implies that the beam search procedure is aligned according to token numbers rather than character positions. In other words, at any given inference step, candidates in the beam may varies greatly in the number of characters generated so far. For instance, one candidate contains 5 characters, whereas another candidate contains 8 characters.

2.4 Length Reward

Our preliminary experiments show that the vanilla approach, as described in Equation 5, produces unsatisfactory results. Detailed analysis shows that the paths explored in the beam search space are dominated by single-character tokens, as shown in Figure 2a. As we all know, multi-character tokens are created by merging characters that frequently occur together, capturing the most common patterns in the language. LLMs are trained for and, in turn, very good at generating multi-character tokens. Therefore, it is counter-intuitive to deprive such capability from LLMs.

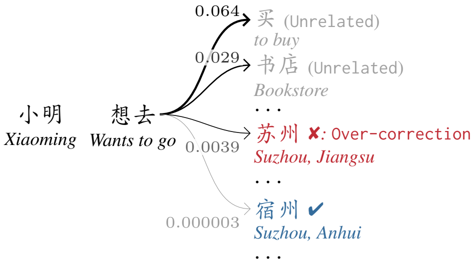


Figure 3: A real example of the probabilities for the next token, given the partial sequence "小明想去" from the sentence "小明想去宿州" (*Xiaoming wants to go to Suzhou, Anhui*).

To handle the issue, we design a simple length reward so that the model favors and keeps multi-char tokens during beam search:

$$\begin{aligned}\mathtt{score}(\boldsymbol{x}, \boldsymbol{t}_{\leqslant j}) = \mathtt{score}(&\boldsymbol{x}, \boldsymbol{t}_{<j}) \\ &+ \log p_{\mathtt{LLM}}(t_j \mid \boldsymbol{t}_{<j}) \\ &+ \sum_{r=1}^{k} \log p_{\mathtt{DM}}(c_r \mid x_{l+r}) \\ &+ \alpha \times (\ell(t_i) - 1)\end{aligned} \tag{6}$$

where α is a hyperparameter for balancing the weight of the length reward, considering that the other two components use log probabilities, whereas the length reward uses numbers directly. Please note that we use $\mathtt{score}(\cdot)$ instead of $p(\cdot)$, since the values are no longer probabilities.

As shown in Figure 2b, thanks to the length reward, the correct token "施工单位" (*construction unit*) is now ranked within the top-K candidates.

2.5 Faithfulness Reward

Under our prompt-free use, the LLM component is unaware of the input sentence, and only focuses on the fluency and correctness of the output sentence from the language use perspective.

We observe that our approach, even with the length reward, tends to over-correct the input sentence, i.e., changing its original meaning. Figure 3 gives an example. Given the partial output sentence, i.e., "小明想去" (*Xiaoming wants to go to*), the LLM component gives a probability of 0.0039 to "苏州" (*sūzhōu*), which is a very famous city in Jiangsu Province. In contrast, it gives a much lower probability of 3×10^{-6} to the original input token, i.e., "宿州" (*sùzhōu*), which is a less famous city in Anhui Province. The distortion model fails to remedy such great gap. As the result, our approach adopts the "correction". However, under such circumstances, it is better to reserve the original tokens.

To mitigate this issue, we introduce a faithfulness reward:

$$\begin{aligned}&\mathtt{score}(\boldsymbol{x}, \boldsymbol{t}_{\leqslant j}) = \mathtt{score}(\boldsymbol{x}, \boldsymbol{t}_{<j}) \\ &\ + \log p_{\mathtt{LLM}}(t_j \mid \boldsymbol{t}_{<j}) \\ &\ + (1 + H_{\mathtt{LLM}(\cdot)}) \times \left(\begin{array}{c} \sum_{r=1}^{k} \log p_{\mathtt{DM}}(c_r \mid x_{l+r}) \\ + \\ \alpha \times (\ell(t_i) - 1) \end{array} \right)\end{aligned} \tag{7}$$

where $H_{\mathtt{LLM}(\cdot)}$ denote the entropy of next-token probabilities.[2] If the entropy is high, meaning that the LLM is uncertain about the next token, the distortion model, along with the length reward, will play a more important role in deciding the next token. From Table 1, we can see that the "Identical" type has a much higher probability than others. That is, the distortion model always favors the original input tokens.

3 Experimental Setup

3.1 Datasets.

Pseudo development set Since there is no publicly available, manually labeled, domain-general development set for CSC, we have chosen to split a small portion of the existing synthetic training data for hyperparameter tuning, naming it **Pseudo-Dev**. Specifically, we use 1,000 sentences each from the synthetic training data of Hu et al. (2022) and Wang et al. (2018) as our development set.

Real-world test sets We perform experiments across five distinct CSC datasets: **Sighans** (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015), **CSCD-IME** (Hu et al., 2022), **MCSCSet** (Jiang et al., 2022), **ECSpell** (Lv et al., 2023), and **Lemon** (Wu et al., 2023), covering a broad spectrum of domains and genres. The details and statistics of these datasets can be found in Appendix B.1. For Sighans, we utilize the revised versions released by Yang et al. (2023b), which have been manually verified and corrected for errors of the original datasets, and name them as **rSighans** for clarity.

Selected datasets for analyses Given the absence of a domain-general development set for CSC and the potential limitations of the **Pseudo-Dev** set in representing real-world data, we conduct

[2] Since LLMs have different output vocabularies $\mathcal{V}$, we divide the entropy by $\log |\mathcal{V}|$, which can be understood as the maximum entropy, and the value will fall into $[0, 1]$.

System		rSighans			CSCD-IME			MCSCSet			ECSpell			Lemon		
		S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓
Domain-Specific SOTAs (*Trained on in-domain gold-standard data of each dataset*)																
ReaLiSe[†]		69.3	80.7	10.1	*41.4*	*44.2*	*27.6*	*17.8*	*27.6*	*12.0*	*34.9*	*45.4*	*13.7*	*28.2*	*31.6*	*19.1*
Hu et al. (2022)		–	–	–	74.4	76.6	–	–	–	–	–	–	–	–	–	–
Jiang et al. (2022)		–	–	–	–	–	–	80.9	–	–	–	–	–	–	–	–
Liu et al. (2023)		–	–	–	–	–	–	–	–	–	85.7	–	5.4	–	–	–
Domain-General SOTAs (*Trained on about 34M synthetic CSC data*)																
Finetuned BERT		47.5	**57.5**	16.9	**52.0**	**53.9**	**25.7**	35.3	48.5	7.5	57.1	64.9	**6.4**	48.0	49.3	13.1
Softmasked BERT		**47.7**	57.4	15.1	51.0	53.4	28.5	35.3	48.5	8.1	57.6	66.2	7.6	47.2	48.8	13.1
ReLM		47.3	56.9	**9.6**	49.5	51.6	29.3	**37.8**	**50.2**	**6.8**	**59.3**	**68.4**	8.6	**50.2**	**51.3**	**11.8**
LLMs (*without CSC-specific training*)																
Baichuan2 (13B)	ZSP	19.0	18.4	49.1	22.6	14.5	35.3	13.6	8.0	77.5	34.5	22.3	30.3	17.5	9.8	40.9
	FSP	31.8	38.5	21.4	35.7	32.7	**10.5**	42.6	47.1	4.4	56.8	53.1	5.8	35.1	25.2	9.5
	OUR	**59.1**	**70.9**	**10.4**	**63.2**	**66.2**	16.5	**66.0**	**76.9**	**1.7**	**84.5**	**89.8**	**4.9**	**53.2**	**56.2**	**9.1**
Qwen1.5 (14B)	ZSP	29.0	31.4	41.1	34.3	31.3	24.5	40.2	45.4	3.8	50.9	49.0	14.4	31.8	26.8	16.1
	FSP	34.3	37.9	26.2	42.9	38.7	**10.4**	40.5	44.3	**3.1**	59.0	58.2	**5.9**	37.2	30.2	**9.9**
	OUR	**54.4**	**68.0**	**17.2**	**52.6**	**57.7**	25.8	**61.1**	**72.6**	**3.1**	**81.6**	**88.2**	6.5	**46.3**	**50.8**	14.1
InternLM2 (20B)	ZSP	31.0	30.4	57.3	34.9	29.2	40.6	19.0	12.5	80.5	45.2	37.5	31.6	32.8	26.5	27.8
	FSP	35.2	38.8	31.7	39.4	35.1	22.4	33.6	32.6	20.4	54.3	49.8	15.7	35.9	28.9	17.3
	OUR	**57.1**	**70.0**	**12.6**	**60.7**	**64.1**	**19.7**	**63.2**	**72.9**	**2.6**	**82.4**	**88.8**	**5.1**	**49.8**	**53.7**	**10.7**

Table 2: Main Results. †: We reran the released code of ReaLiSe (Xu et al., 2021), along with their released models, to obtain the results. ReaLiSe, was trained on the in-domain, gold-standard data of the Sighans dataset and represents a SOTA model for it. The numbers in *gray* represent the out-of-domain results for ReaLiSe.

in-depth analyses on three distinct datasets to cover a broad spectrum of language use. These include errors made by Chinese learners (**rSighan** *15*), colloquial and diverse text from novels (**Lemon** *Nov*), and formal and standard text from official documents (**ECSpell** *Odw*).

3.2 Evaluation Metrics.

We follow the convention to use the **sentence-level correction** F_1 (**S-F**) score as the main evaluation metric. Besides, we also report **character-level correction** F_1 (**C-F**) and **sentence-level false positive rate** (**FPR**) to provide a more complete view of the model performance.

3.3 Baselines

We compare our approach against prompt-based method under two settings: zero-shot prompting (**ZSP**) and few-shot prompting (**FSP**). For few-shot settings, we select 10 examples from the **Pseudo-Dev**. The details of the prompts can be found in Appendix B.4, and the example selection strategy is described in Appendix B.5. During inference, we adopt the greedy decoding strategy.[3]

We do not compare against supervised fine-tuning methods (SFT) in this study for two reasons. First, our approach is training-free, making direct comparisons with SFT unfair. Second, SFT are computationally expensive and time-consuming, particularly for LLMs.

To provide a more comprehensive comparison, we also present results from state-of-the-art domain-general CSC models trained on 34 million pairs of synthetic CSC data for reference. These models include **Finetuned BERT** (Devlin et al., 2019), **Softmasked BERT** (Zhang et al., 2020), and **ReLM** (Liu et al., 2023).[4]

Additionally, for datasets that have in-domain manually annotated data, we report results from models specifically trained on it, serving as another reference point.

3.4 Selection of LLMs

We conduct experiments on three open-source LLMs: Baichuan2 (Yang et al., 2023a), Qwen1.5 (Bai et al., 2023), and InternLM2 (Cai et al., 2024). For the main results, we select models with parameter sizes ranging from 10B to 20B to ensure that the LLMs have sufficient zero-shot and few-shot capabilities for meaningful comparisons. Additionally, we report the ZSP and FSP results of the

[3] We observe that the improvement of beam search is marginal and sometimes even detrimental.

[4] The results of these models were obtained by running the released code along with the corresponding checkpoints provided at https://github.com/gingasan/lemon.git.

System			**S-F**$\uparrow$	S-P$\uparrow$	S-R$\uparrow$	**C-F**$\uparrow$	C-P$\uparrow$	C-R$\uparrow$	**FPR**$\downarrow$
rSighan *15*									
ReLM			55.5	61.1	50.8	61.0	78.5	49.9	9.5
GPT3.5		ZSP	42.0	41.7	42.3	44.6	39.7	50.8	25.9
		FSP	41.7	42.0	41.4	45.4	41.7	49.9	23.4
GPT4		ZSP	43.5	38.1	50.8	47.1	37.9	62.2	47.5
		FSP	48.7	44.2	54.4	50.6	42.1	**63.4**	38.8
BC2	13B	OUR	59.6	66.5	54.0	67.3	78.3	59.0	**8.3**
Q1.5	14B		57.6	62.5	53.4	66.0	74.1	59.4	10.2
IL2	20B		**60.5**	**67.2**	**55.0**	**67.8**	**78.7**	59.6	**8.3**
Lemon *Nov (1000)*									
ReLM			36.4	46.7	29.8	36.0	49.2	28.3	14.3
GPT3.5		ZSP	19.2	20.8	17.9	19.6	17.5	22.2	29.8
		FSP	25.5	31.4	21.4	24.0	26.2	22.2	19.6
GPT4		ZSP	30.6	28.4	33.1	31.9	25.2	43.4	33.5
		FSP	42.7	41.4	**44.0**	42.2	38.1	**47.3**	27.4
BC2	13B	OUR	**45.3**	**53.7**	39.1	**49.1**	**57.0**	43.2	**13.1**
Q1.5	14B		38.2	41.7	35.3	43.7	44.5	43.0	21.8
IL2	20B		42.8	49.9	37.5	46.4	52.8	41.4	15.3
ECSpell *Odw*									
ReLM			66.5	67.5	65.6	73.0	86.4	63.1	7.1
GPT3.5		ZSP	57.7	61.9	54.1	59.1	60.4	57.8	4.9
		FSP	59.3	64.1	55.2	59.8	61.6	58.2	2.4
GPT4		ZSP	73.1	73.0	73.3	75.6	73.8	77.5	5.0
		FSP	73.2	73.5	72.9	78.0	77.8	78.2	5.0
BC2	13B	OUR	**92.0**	**94.4**	**89.7**	**93.8**	95.6	**92.1**	**0.4**
Q1.5	14B		87.4	88.6	86.3	91.6	91.8	91.3	2.9
IL2	20B		91.1	92.9	89.3	**93.8**	**95.9**	91.8	**0.4**

Table 3: The comparison to GPT family on the rSighan 15, Lemon Nov, and ECSpell Odw datasets. The version of GPT3.5 is 'gpt-3.5-turbo-0125', GPT4 is 'gpt-4-0613'. BC2 is short for Baichuan2, Q1.5 for Qwen1.5, and IL2 for InternLM2.

widely recognized best-performing LLM family, GPT, including GPT-3.5 and GPT-4.

To simplify the analysis, we select the Baichuan2 7B as a representative model to investigate the impact of components in our approach.

3.5 Hyperparameters of Our Approach

We use the "Base" version of each LLM family. The distortion probabilities of distortion model were derived from the statistics of the Pseudo-Dev dataset. We tuned α on Baichuan2 7B using the Pseudo-Dev dataset. Eventually, α was set to 2.5 for all experiments. During inference, we adopt beam search with a beam size of 8.

4 Main Results

We present the main results in Table 2, and the comparison to the GPT family in Table 3. Conducting a comprehensive evaluation of the GPT family is expensive, so we limit the comparison to a small-scale study, focusing on the three datasets mentioned in Section 3.1.[5] Moreover, two qualitative examples are provided in Appendix C.2 to illustrate the performance of our approach.

After applying our approach, all three LLM families outperforms their zero-shot and few-shot prompting counterparts on all five datasets by a large margin.

Compared to the recent state-of-the-art domain-general CSC models, which are trained on 34M synthetic CSC data, our approach also achieves competitive or even superior performance on most datasets, especially on the MCSCSet and ECSpell datasets. The results indicate that our approach has a better generalization across different domains and genres than the current domain-general SOTAs. However, our approach still largely lags behind the domain-specific SOTAs trained on the gold-standard labeled data (from 1.2 to 21.8 on S-F score) of each dataset.

Compared to the GPT family, our approach consistently outperforms GPT3.5 on all three datasets, and achieves better performance than GPT4 in most cases. However, our approach may exhibit a lower recall rate for character-level corrections compared to GPT4, indicating that we might miss some errors that GPT4 can successfully correct.

5 Discussion

5.1 Impact of the Size of the LLM

First, we investigate the impact of the LLM size on the performance of our approach.

As shown in Table 4, in general, larger LLMs tend to perform better than smaller ones within the same model family. However, the Qwen1.5 model family is an exception: the performance improvement becomes marginal when the model size exceeds 1.8B parameters and even decreases when the model size reaches 7B.

When comparing the performance of models of the same size across different model families, we find that the Baichuan2 family generally outperforms the other two model families.

5.2 Effectiveness of the Minimal Distortion Model

To investigate the effectiveness of the minimal distortion model, we first remove the distortion model

[5] The original Lemon-Nov dataset includes 6,000 sentences, which is excessively large for our scope. Therefore, we selected the first 1,000 sentences for this comparison.

Model	Size	rSighan *15*			Lemon *Nov*			ECSpell *Odw*		
		S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓
`Baichuan2`	`7B`	**59.8**	**68.2**	**8.0**	43.2	47.7	13.6	89.7	93.0	1.3
	`13B`	59.6	67.3	8.3	**43.5**	**47.9**	**13.0**	**92.0**	**93.8**	**0.4**
`Qwen1.5`	`0.5B`	56.3	63.5	10.0	33.2	40.2	22.2	84.7	89.9	3.8
	`1.8B`	58.3	65.3	10.3	35.6	42.3	19.9	**90.3**	**92.8**	**1.7**
	`4B`	58.4	66.8	10.0	35.9	42.3	21.1	88.4	91.1	3.4
	`7B`	**59.4**	**67.0**	**8.5**	**39.0**	**44.7**	**19.0**	87.1	91.4	3.4
	`14B`	57.6	66.0	10.2	36.4	42.6	21.2	87.4	91.6	2.9
`InternLM2`	`1.8B`	55.3	64.0	12.2	33.2	40.1	22.6	88.3	91.0	2.1
	`7B`	58.1	65.5	10.2	38.8	44.2	18.0	89.3	92.0	2.1
	`20B`	**60.5**	**67.8**	**8.3**	**40.5**	**45.3**	**15.1**	**91.1**	**93.8**	**0.4**

Table 4: Ablation results of model size.

System	**S-F↑**	S-P↑	S-R↑	**C-F↑**	C-P↑	C-R↑	**FPR↓**
rSighan *15*							
`CTG`	6.7	5.3	9.1	7.7	4.2	47.7	90.0
`OUR`	59.8	66.0	54.7	68.2	77.8	60.6	8.0
`- DT`	-7.7	-12.6	-3.9	-7.1	-15.7	-0.3	+9.4
`- DT`†	-12.3	-18.2	-7.5	-9.8	-20.5	-1.2	+11.1
Lemon *Nov*							
`CTG`	0.7	0.5	1.1	1.4	0.7	22.5	96.2
`OUR`	43.2	52.2	36.9	47.7	55.5	41.9	13.6
`- DT`	-12.3	-20.5	-6.8	-10.0	-20.8	-0.7	+13.9
`- DT`†	-11.7	-20.5	-5.5	-9.7	-21.8	-1.6	+14.7
ECSpell *Odw*							
`CTG`	29.3	24.5	36.3	21.4	12.4	79.5	52.9
`OUR`	89.7	91.6	87.8	93.0	95.3	90.8	1.3
`- DT`	-4.0	-4.6	-3.4	-3.9	-5.8	-2.2	0.0
`- DT`†	-16.3	-16.9	-15.7	-12.7	-14.5	-10.9	+2.5

Table 5: Ablation results of distortion model on `Baichuan2 7B`. "`CTG`" means constrained text generation. "`-DT`" represents that we do not distinguish `Same Pinyin`, `Similar Pinyin`, and `Similar Shape`, and treat them as `Related` distortion. "`-DT`†" represents that using the confusion set from Wang et al. (2018) to identify the `Related` distortion.

$p_{\texttt{DM}}(\boldsymbol{x} \mid \boldsymbol{y})$ from the decoding process. Alternatively, we adopt a constrained text generation (CTG) approach to correct the input sentence. For each step, we limit the vocabulary to tokens that are related to the corresponding characters in the input sentence,[6] and let the model select the most likely token from the constrained vocabulary. The results are shown in the "`CTG`" column in Table 5. We can see that the CTG performs poorly on all datasets. This is because a Chinese character may have many similar characters. Without the distortion model, the model is prone to replacing the original character with a higher-frequency similar character, leading to a large number of errors.

Next, we investigate the impact of the distortion type by treating three types of related but not identical distortions as a single distortion type. As shown in the "`- DT`" column in Table 5, the performance drops significantly but not as severely as when removing the distortion model. This performance drop is mainly due to a decrease in precision.

We also examine the effectiveness of our rule-based tool for identifying related distortions. We replace our rule-based tool with the confusion set from Wang et al. (2018) to identify the related distortion. The results in the "`- DT`†" column in Table 5 show that the confusion set from Wang et al. (2018) is less effective than our rule-based tool, leading to more severe performance degradation.

Moreover, considering that the estimated distortion probabilities may differ from the true ones, an analysis to verify the effectiveness of the estimated distortion model is provided in Appendix D.2.

5.3 Impact of two Rewards

In this work, we propose two rewards to optimize the decoding process: the length reward and the faithfulness reward. The ablation study results of the two rewards are shown in Table 6.

The results show that the length reward significantly improves performance on all three datasets. This improvement can be attributed to increases in both precision and recall, indicating that the length reward is crucial to our approach.

The results demonstrate that the faithfulness reward can effectively improve precision, although it may slightly reduce recall. Overall, the faithfulness reward balances the trade-off between precision and recall, leading to a higher correction F_1 score.

The combination of the two rewards can achieve better performance than using them separately, especially when the datasets contain less formal

[6] Classified as `Identical`, `Same Pinyin`, `Similar Pinyin`, or `Similar Shape`.

System	S-F↑	S-P↑	S-R↑	C-F↑	C-P↑	C-R↑	FPR↓
rSighan *15*							
Vanilla	18.0	15.9	20.6	20.7	14.3	37.6	52.9
w/ LR	+39.4	+43.4	+35.0	+43.7	+53.3	+23.9	-38.4
w/ FR	+3.8	+6.2	+0.8	+5.4	+8.3	-6.6	-19.3
w/ Both	+41.9	+50.1	+34.1	+47.4	+63.5	+23.0	-44.8
Lemon *Nov*							
Vanilla	19.4	18.0	20.9	23.6	17.1	38.3	38.5
w/ LR	+17.1	+19.5	+14.6	+19.0	+21.9	+8.6	-13.7
w/ FR	+9.0	+13.5	+4.7	+8.5	+13.5	-4.5	-18.8
w/ Both	+23.9	+34.2	+16.0	+24.1	+38.4	+3.6	-25.0
ECSpell *Odw*							
Vanilla	65.3	65.3	65.3	70.4	65.4	76.2	10.1
w/ LR	+25.4	+26.9	+24.0	+22.5	+28.5	+15.6	-9.7
w/ FR	+4.7	+11.2	-0.8	+7.5	+19.7	-4.5	-6.7
w/ Both	+24.4	+26.4	+22.5	+22.6	+29.9	+14.6	-8.8

Table 6: Ablation results of Baichuan2 7B. "LR" and "FR" represent "length reward" and "faithfulness reward" respectively. "Both" means using both length reward and faithfulness reward.

text, more colloquial expressions, and more diverse named entities, such as the rSighan-15 and Lemon-Nov datasets.

6 Related Works

6.1 Chinese Spelling Check

Previous research on the CSC task can be divided into three eras, accompanied with paradigm shift.

The Early Unsupervised Era Early CSC approaches mainly utilized unsupervised pipeline systems (Yeh et al., 2013; Yu et al., 2014; Yu and Li, 2014; Huang et al., 2014; Xie et al., 2015). These systems typicaly act in three main steps: error detection, candidate correction generation from a confusion set, and candidate ranking using a statistical n-gram language model.

The Supervised Learning Era By 2018, the advent of techniques for automatically generating pseudo-labeled data had begun to address the challenge of data scarcity in CSC (Wang et al., 2018), marking a shift in the paradigm of CSC research towards a supervised learning era dominated by deep neural networks. This era saw researchers exploring various avenues to enhance CSC performance. Some focused on finding better model architectures (Zhang et al., 2020; Zhu et al., 2022), while others delved into more effective training strategies (Liu et al., 2022; Wu et al., 2023; Liu et al., 2023). Additionally, there was an effort to enrich models with information beyond text, such as phonetic or visual features (Cheng et al., 2020; Xu et al., 2021; Li et al., 2022; Liang et al., 2023).

Similar to our work, Wu et al. (2023) also decomposed $p(\boldsymbol{x} \mid \boldsymbol{y})$ into two parts to improve CSC performance. However, they achieved this by adding an *auxiliary training loss*. Our work stands out by using an off-the-shelf LLM as the backbone and a minimal distortion model to achieve good CSC performance without any additional training.

The LLM Era Our work represents an initial foray into what could be considered the third era of CSC research: the LLM era. This phase explores the potential of LLMs in addressing the CSC task. As mentioned in the introduction, related studies in this era fall into two categories: *prompt-based* and *supervised fine-tuning*. In contrast to these methods, our approach requires neither additional training nor prompts.

6.2 Decoding Methods of LLMs

Intervening in the decoding process is a common approach to improve LLMs' task-specific performance. There are two popular approaches in this category: **Contrastive decoding** and **Constrained decoding**. Contrastive decoding (Li et al., 2023b) refines the output probabilities by comparing the output probabilities of expert and amateur models (O'Brien and Lewis, 2023; Shi et al., 2023). Constrained decoding, on the other hand, uses constraints to guide the decoding process, making the output more aligned with the task-specific requirements (Wang et al., 2023; Geng et al., 2023).

Our work is closely related to the constrained decoding approaches, where a distortion model is used to influence the LLM decoding process.

7 Conclusion

In this work, we propose a simple, training-free, and prompt-free approach to leverage LLMs for the CSC task. Two components, a large language model and a minimal distortion model, co-operate to correct spelling errors. We alleviate the local optima problem and over-correction issue, with two simple strategies, length reward and faithfulness reward, respectively. Our comprehensive experiments have shown that our approach significantly improves LLM performance. Through our approach, LLMs demonstrate remarkable domain generalization capabilities, surpassing SOTA domain-general CSC models, that are trained on extensive synthetic CSC data, on most datasets.

Limitations

The scope of this study is limited to the task of Chinese spelling correction, which is a subset of text error correction. Most of our design choices are tailored to the characteristics of Chinese and the specific requirements of the CSC task.

However, our approach has the potential to be directly applied to some other languages. For example, in Japanese and Korean, we can also categorize errors into phonetic similarities, such as (や, *ya*)-(な, *na*) in Japanese or (후, *hu*)-(부, *bu*) in Korean, and shape similarities, like (ユ, *yu*)-(エ, *e*) in Japanese. For languages using a phonetic writing system, like English, minor adjustments such as adding `INSERT`, `DELETE`, and `REORDER` operations will be sufficient to make it work.

Comparatively, handling complex text errors that involve grammar, semantics, or pragmatics, are more challenging. To tackle these errors, one could design an appropriate distortion model, though it might necessitate the adoption of more intricate rules or the implementation of a model based on neural networks. In our future work, we aim to explore ways that would allow our approach to handle these complex errors.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *ArXiv preprint*, abs/2309.16609.

Zuyi Bao, Chen Li, and Rui Wang. 2020. Chunk-based Chinese spelling check with global optimization. In *Proceedings of EMNLP*, pages 2031–2040, Online.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *ArXiv preprint*, abs/2403.17297.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, Minneapolis, Minnesota.

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*, abs/2403.08492.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of EMNLP*, pages 10932–10952, Singapore.

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 W-NUT*, pages 160–169, Hong Kong, China.

Yong Hu, Fandong Meng, and Jie Zhou. 2022. CSCD-IME: correcting spelling errors generated by pinyin IME. *ArXiv preprint*, abs/2211.08788.

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.

Wangjie Jiang, Zhihao Ye, Zijing Ou, Ruihui Zhao, Jianguang Zheng, Yi Liu, Bang Liu, Siheng Li, Yujiu Yang, and Yefeng Zheng. 2022. Mcscset: A specialist-annotated dataset for medical-domain Chinese spelling correction. In *Proceedings of CIKM*, pages 4084–4088.

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.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023b. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of ACL*, pages 12286–12312, Toronto, Canada.

Yinghui Li, Haojing Huang, Shirong Ma, Yong Jiang, Yangning Li, Feng Zhou, Hai-Tao Zheng, and Qingyu Zhou. 2023a. On the (in)effectiveness of large language models for Chinese text correction. *ArXiv preprint*, abs/2307.09007.

Zihong Liang, Xiaojun Quan, and Qifan Wang. 2023. Disentangled phonetic representation for Chinese spelling correction. In *Proceedings of ACL*, pages 13509–13521, Toronto, Canada.

Linfeng Liu, Hongqiu Wu, and Hai Zhao. 2023. Chinese spelling correction as rephrasing language model. *ArXiv preprint*, abs/2308.08796.

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 ACL*, pages 3008–3018, Dublin, Ireland.

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*, 22(5).

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models. *ArXiv preprint*, abs/2309.09117.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen-tau Yih. 2023. Trusting your evidence: Hallucinate less with context-aware decoding. *ArXiv preprint*, abs/2305.14739.

Chenglei Si, Zhengyan Zhang, Yingfa Chen, Xiaozhi Wang, Zhiyuan Liu, and Maosong Sun. 2023. READIN: A Chinese multi-task benchmark with realistic and diverse input noises. In *Proceedings of ACL*, pages 8272–8285, Toronto, Canada.

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.

Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A. Saurous, and Yoon Kim. 2023. Grammar prompting for domain-specific language generation with large language models. *ArXiv preprint*, abs/2305.19234.

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.

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 *Proceedings 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. *ArXiv preprint*, abs/2309.10305.

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*, 4:183–192.

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.

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.

Chenxi Zhu, Ziqiang Ying, Boyu Zhang, and Feng Mao. 2022. MDCSpell: A multi-task detector-corrector framework for Chinese spelling correction. In *Findings of ACL*, pages 1244–1253, Dublin, Ireland.

A Implement of Distortion Model

A.1 Standard of Transformation Types

Identical Transformations An identical distortion occurs when the input character is the same as the correct character.

Same Pinyin Characters that share the same pronunciation, disregarding tone, undergo a "`Same Pinyin`" distortion. Due to the existence of heteronyms in Chinese, such as "和", which can be pronounced in multiple ways including "*hé*", "*hè*", "*huó*", "*huò*", and "*hú*", we classify two characters as undergoing a same pinyin distortion if they share at least one pronunciation. The `pypinyin`[7] library is utilized to determine character pronunciations, with the `ktghz2013` and `large_pinyin` from `pypinyin-dict`.[8] providing a more accurate pronunciation for these determinations.

Similar Pinyin We categorize distortions as "`Similar Pinyin`" when two characters have pronunciation that is recognized as similar by predefined rules, which are based on Yang et al. (2023b). For instance, '*qī*" and "*jī*" are considered similar due to the common mispronunciation of the consonant "*q*" as "*j*". A list of consonants and vowels considered similar can be found in Tables 7 and 8, respectively.

Similar Shape The similarity in the shape of characters is evaluated by combining their four-corner code with their radical and component information. For example, the characters "机" and "仉" have the four-corner codes "`47910`" and "`27210`", respectively. Given that the last digit primarily serves to distinguish characters with identical preceding digits and that "机" and "仉" share two of these digits, their four-corner code similarity is calculated as $2 \times \frac{1}{4} = 0.5$. Considering their radical and component ("木, 几" for "机" and "人, 几" for "仉"), which share the component "几" but differ in radicals, their similarity is $1 \times \frac{1}{2} = 0.5$. Thus, the overall similarity is averaged to 0.5. With a similarity threshold set at 0.45, these characters are considered to undergo a similar shape distortion. Furthermore, character pairs where one is a radical or component of the other, such as "机" and "几", are also classified under similar shape distortions.

All non-Chinese characters are only allowed to be transformed into themselves.

Corrected → Input										
j	→	●	*q*	*x*	*z*	☒	☒	☒	☒	☒
q	→	*j*	●	*x*	☒	*c*	☒	☒	☒	☒
x	→	*j*	*q*	●	☒	☒	*s*	☒	☒	☒
z	→	*j*	☒	☒	●	*c*	*s*	*zh*	☒	☒
c	→	☒	*q*	☒	*z*	●	*s*	☒	*ch*	☒
s	→	☒	☒	▩	*z*	*c*	●	☒	☒	*sh*
zh	→	☒	☒	☒	*z*	☒	☒	●	*ch*	*sh*
ch	→	☒	☒	☒	☒	*c*	☒	*zh*	●	*sh*
sh	→	☒	☒	☒	☒	☒	*s*	*zh*	*ch*	●
r	→	●	*l*	☒	☒	☒	☒	☒	☒	
l	→	*r*	●	*n*	*d*	*t*	☒	☒	☒	
n	→	☒	*l*	●	*d*	*t*	☒	☒	☒	
d	→	☒	*l*	*n*	●	*t*	*b*	☒	☒	
t	→	☒	*l*	*n*	*d*	●	☒	*p*	☒	
b	→	☒	☒	☒	*d*	☒	●	*p*	*m*	
p	→	☒	☒	☒	☒	*t*	*b*	●	▩	
m	→	☒	☒	☒	☒	☒	*b*	*p*	●	
g	→	●	*k*	*h*	☒					
k	→	*g*	●	*h*	☒					
h	→	*g*	*k*	●	*f*					
f	→	☒	☒	*h*	●					

Table 7: Consonants with similar pronunciation.

Corrected → Input							
an	→	●	*ang*	*uan*	*uang*	*ian*	☒
ang	→	*an*	●	*uan*	*uang*	☒	*iang*
uan	→	*an*	*ang*	●	*uang*	*ian*	☒
uang	→	*an*	*ang*	*uan*	●	☒	*iang*
ian	→	*an*	☒	*uan*	☒	●	*iang*
iang	→	☒	*ang*	☒	*uang*	*ian*	●
en	→	●	*eng*	*un*	☒		
eng	→	*en*	●	☒	☒		
un	→	*en*	☒	●	*ong*		
ong	→	☒	☒	*un*	●		
in	→	●	*ing*				
ing	→	*in*	●				
o	→	●	*uo*				
uo	→	*o*	●				
ü	→	●	*u*				
u	→	*ü*	●				

Table 8: Vowels with similar pronunciation.

A.2 Type Priority

In scenarios where a character can be classified under multiple distortion types, for example, "机" (*jī*) and "玑" (*jī*), which can be classified as both having

[7] https://github.com/mozillazg/python-pinyin

[8] https://github.com/mozillazg/pypinyin-dict

Datasets	**rSighans**			**CSCD**	**MCSC**	**ECSpell**		
Subsets	Y13	Y14	Y15	Test	Test	Law	Med	Odw
#Sentence	1,000	1,062	1,100	5,000	19,650	500	500	500
Erroneous Sentence Ratio	97.70	56.69	56.18	46.06	50.00	51.00	45.20	52.40
Average Length	74.33	50.01	30.64	57.63	10.91	29.74	49.60	40.51
Average Error/Sentence	1.48	0.88	0.78	0.51	0.93	0.78	0.71	0.81
Distortion Type Proportion (%)								
`Identical`	98.01	98.25	97.45	99.12	91.47	97.38	98.56	98.01
`Same Pinyin`	1.62	1.30	1.83	0.74	6.60	1.82	1.15	1.55
`Similar Pinyin`	0.28	0.40	0.66	0.13	1.05	0.51	0.19	0.28
`Similar Shape`	0.05	0.01	0.03	0.00	0.39	0.25	0.08	0.13
`Unrelated`	0.04	0.04	0.02	0.00	0.45	0.04	0.01	0.02
Recall Upper Bound	97.24	97.18	98.71	99.70	90.82	97.65	98.67	98.47

Datasets	**Lemon**							**Pseudo-Dev**
Subsets	Car	Cot	Enc	Gam	Med	New	Nov	–
#Sentence	3,410	1,026	3,434	400	2,090	5,892	6,000	2,000
Erroneous Sentence Ratio	51.09	46.20	50.99	38.75	50.38	50.00	50.23	93.55
Average Length	43.44	40.12	39.83	32.99	39.28	25.16	36.24	36.94
Average Error/Sentence	0.56	0.47	0.52	0.41	0.49	0.55	0.57	1.42
Distortion Type Proportion (%)								
`Identical`	98.64	98.78	98.63	98.73	98.64	97.80	98.43	96.15
`Same Pinyin`	0.90	0.75	0.93	0.89	0.94	1.50	0.95	2.34
`Similar Pinyin`	0.31	0.25	0.28	0.26	0.27	0.51	0.43	0.78
`Similar Shape`	0.02	0.07	0.06	0.01	0.02	0.05	0.02	0.40
`Unrelated`	0.12	0.14	0.09	0.11	0.12	0.13	0.16	0.31
Recall Upper Bound	91.38	89.34	94.28	90.54	92.82	93.98	89.08	88.03

Table 9: The statistics of the datasets used in the experiments. **Recall Upper Bound** represents the sentence-level upper bound of the recall under the distortion model that we use in this work.

the same pinyin and a similar shape, we prioritize the distortion type according to the following order: 1) `Identical`; 2) `Same Pinyin`; 3) `Similar Pinyin`; 4) `Similar Shape`; 5) `Unrelated`.

A.3 Using an Inverted Index for Efficient Distortion Model Calculation

During each decoding step, the distortion model calculates the probability of transforming the input sequence $x_{a:b}$ into a candidate token t_i:

$$g(x, t_i) = \sum_{r=1}^{k} \log p_{\texttt{DM}}(c_r \mid x_{l+r}), \qquad (8)$$

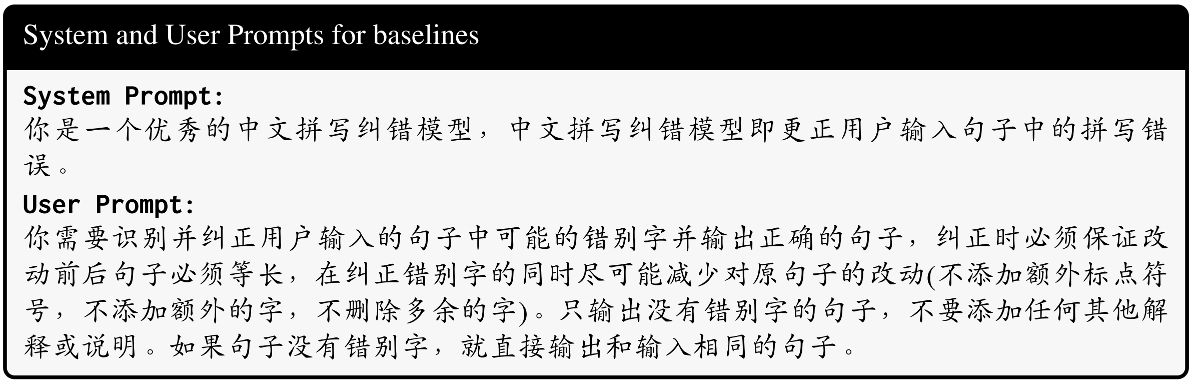
where the function $g(x, t_i)$ must be computed for each candidate token t_i in the vocabulary $\mathcal{V}$, resulting in a huge computational cost.

To address this challenge, we propose the use of an inverted index to reduce the calculation process, by only considering relevant tokens, and ignoring irrelevant tokens. For a token, we can preconstruct indexed entries to represent it, such as `<0,ji,SamePinyin>`, `<1,kou,SimilarPinyin>`, and `<0,仉,SimilarShape>` for “机构” (*jī gòu*). Upon receiving an input sequence, the index enables rapid retrieval of relevant tokens, thereby limiting probability calculations exclusively to these tokens. As the subset of relevant tokens is substantially smaller than the complete token set, employing an inverted index considerably reduces the computational burden.

A.4 Small Tricks for Distortion Model

We adopt three small tricks to enhance our distortion model. First, for character pairs commonly misused in everyday writing, such as “的”, “地”, and “得”, we categorize these as “`Identical`” distortions, allowing the model to correct these errors with lower difficulty.

Second, we found that, although the previously described rules adequately cover most similar relationships between characters, a few exceptions, approximately 0.01% of total character pairs, still



System and User Prompts for baselines

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必须等长，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 our `FSP` and `ZSP` baselines.

persist. To identify these outliers, we leveraged tools from previous studies (Wu et al., 2023; Hu et al., 2022) by incorporating their structure confusion sets and spelling similarity matrices. We classify character pairs found within the structure confusion set or those with a spelling similarity matrix distance of less than 1 as "`Other Similar`" distortions.

Finally, we have chosen not to entirely exclude unrelated distortions. Instead, we allow each token to possess up to one unrelated character distortion, to which we assign a very low probability.

Employing these tricks has led to marginal yet consistent improvements in our approach's performance.

B Details of Experiments

B.1 Details of Real-world Test Sets

This section details the test sets used in our study, providing insights into their composition and relevance to real-world Chinese text.

- **Sighan series**: This series of datasets is one of the most widely used benchmark datasets for Chinese spelling correction (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015). However, it faces criticism for two main reasons: firstly, it consists of essays written by Chinese learners, which may not accurately represent typical Chinese texts. Secondly, its limited diversity could hinder the evaluation of models' generalization capabilities. Despite these concerns, we include it in our evaluation to allow for comparison with prior studies. However, we utilize the revised version by Yang et al. (2023b), which has manually verified and corrected the errors in the original dataset.
- **CSCD-IME**: A real-world Chinese social media corpus collected and annotated by Hu et al. (2022). It can better represent the variety of texts found in real-world settings and includes a broad spectrum of errors.
- **MCSCSet**: A large-scale corpus from the medical domain, collected and annotated by Jiang et al. (2022). It features numerous errors specific to medical terminology, making it an excellent resource for evaluating models' generalization capabilities in this area.
- **ECSpell**: A small-scale, multi-domain corpus annotated by Lv et al. (2023). It encompasses three domains: legal documents, medical treatments, and official document writing.
- **Lemon**: The most recent and largest multi-domain corpus to date, collected and annotated by Wu et al. (2023). It spans seven domains: law, medicine, encyclopedia, gaming, automotive, contracts, news, and novels. The original dataset also includes sighan 15 as a subset, which we have considered as a part of the Sighan series and excluded from Lemon.

The detailed statistics of these datasets are shown in Table 9.

The recall upper bound in the statistics is obtained by calculating the number of sentences that can potentially be fully corrected out of the total number of sentences in the dataset. A sentence has the potential to be fully corrected if all the distortion types between each pair of source and target characters can be categorized into `Identical`, `Same Pinyin`, `Similar Pinyin`, and `Similar Shape`.

B.2 Evaluation Details

During evaluation, we convert all full-width punctuation to half-width and remove all whitespaces from the input and output sentences to guarantee a fair comparison. When evaluating the Lemon dataset, we ignore all sentences where the input and output sentence lengths do not match, following the dataset's convention.

B.3 Levenshtein Alignment for Character-Level Evaluation

Traditional point-wise evaluation methods fall short when models insert or delete characters, as they can inaccurately mark all subsequent characters as incorrect due to a single addition or deletion. To overcome this, we implement Levenshtein algorithm to align the model output with the target sentence. This approach allows us to calculate character-level metrics based on the aligned results, providing a more reasonable evaluation of character-level performance.

B.4 Implementation Details of Prompt-based Method

In this work, we use the prompt-based method to activate the CSC ability of the baseline LLMs. The task-specific instructions are adopted from Li et al. (2023a). The prompt used for the baselines are shown in Table 4. We disable the sampling mechanism and set the temperature to 0.0 to ensure deterministic decoding. For few-shot prompting methods, where the example selection strategy involves random selection, we conduct three runs and report the average results. The only exception is the GPT4 model, which we run only once due to the high cost of using the model.

B.5 Few-shot Examples Selection Strategy for Baselines

Li et al. (2023a) proposed three selection strategies for CSC few-shot prompting methods: 1) Random: randomly select m examples; 2) Balanced: randomly select m examples with a balanced distribution of correct and error examples; 3) Similarity: select the m most similar in-context examples for each input sentence using the BM25 and Rouge similarity metrics.

They found that the performance of few-shot prompting depends on the selection of in-context examples. Different selection strategies may lead to distinct results. Among the three strategies, Similarity was found to be the most effective.

However, the Similarity strategy is not always the optimal choice. In preliminary experiments, we observed that this strategy sometimes causes GPT family models to perform worse than the zero-shot prompting method. Upon analyzing the results, we found that GPT models are particularly sensitive to discrepancies in the proportion of erroneous sentences between the few-shot prompting examples and the target data. The examples selected using the Similarity strategy tend to have a similar proportion of erroneous sentences as the dataset used for selection. In our work, we use Pseudo-Dev dataset to select few-shot prompting examples, which contains a higher proportion of erroneous sentences (87%–94%) compared to the target data (50%–56%). This discrepancy causes the GPT models to be more aggressive in correcting errors.

To ensure the effectiveness of the few-shot prompting method, we conducted experiments to determine the optimal strategy for each LLM we used. For open-source LLMs, which include both 'Base' and 'Chat' versions, we experimented with both versions and selected the best one for each LLM. The final choice of selection strategy is shown in Table 10.

Model	Version	Strategy
Baichuan2 13B	Base	Similariy
Qwen1.5 14B	Base	Balanced
InternLM2 20B	Chat	Similariy
GPT3.5	–	Balanced
GPT4	–	Balanced

Table 10: The model version and examples selection strategy we used for few-shot baseline.

B.6 Pre- & Post-processing for Baselines

In this study, we employ several pre- and post-processing techniques to mitigate the errors introduced by the limitations of baseline systems. This ensures a fair comparison between our approach and the baselines.

BERT-based baselines Most current CSC models utilize BERT as the backbone. However, BERT presents challenges that can degrade performance during evaluation: 1) *Full-width Punctuation:* BERT's tokenization process may normalize full-width punctuation to half-width, leading to numerous unnecessary punctuation replacements. To counter this, we prevent the model from modifying the original punctuation; 2) *Special Tokens:* BERT-based models may predict a special '[UNK]' token in some cases, resulting in the removal of the original character. In these instances, we retain the original character when a special token is predicted; 3) *Input Length Limitation:* BERT-based

Datasets		**rSighans**			**ECSpell**			**Lemon**						
Subsets		Y13	Y14	Y15	Law	Med	Odw	Car	Cot	Enc	Gam	Med	New	Nov
Domain-Specific SOTAs (*Trained on in-domain gold-standard data of each dataset*)														
ReaLiSe		70.1	64.0	73.9	*38.9*	*23.1*	*42.8*	*32.5*	*40.1*	*29.1*	*12.6*	*31.8*	*31.2*	*20.2*
Liu et al. (2023)		–	–	–	91.2	82.4	83.6	–	–	–	–	–	–	–
Domain-General SOTAs (*Trained on about 34M synthetic CSC data*)														
Finetuned BERT		50.6	40.4	51.6	58.5	47.8	65.1	52.0	63.1	45.3	32.8	50.7	56.1	35.8
Softmasked BERT		**51.6**	40.2	51.3	58.5	48.5	65.9	52.3	63.8	44.1	28.3	48.9	55.6	**37.7**
ReLM		45.8	**40.6**	**55.5**	**60.4**	**50.9**	**66.5**	**53.3**	**66.7**	**47.7**	**33.7**	**53.8**	**58.8**	37.1
LLMs (*without CSC-specific training*)														
Baichuan2 (13B)	ZSP	26.4	12.0	18.5	37.6	23.0	43.0	15.3	14.9	24.0	12.7	21.6	19.8	14.1
	FSP	41.1	23.1	31.3	60.2	50.4	60.0	32.2	45.3	38.9	24.6	39.0	39.7	26.4
	OUR	**63.6**	**54.1**	**59.6**	**82.6**	**78.9**	**92.0**	**52.7**	**62.9**	**51.9**	**37.1**	**60.1**	**63.9**	**43.5**
Qwen1.5 (14B)	ZSP	41.6	17.4	28.1	53.3	38.9	60.7	28.5	42.0	33.8	20.5	35.3	37.3	25.3
	FSP	45.9	25.4	31.6	61.4	49.1	66.5	35.0	47.6	43.4	27.9	38.6	38.7	29.2
	OUR	**56.9**	**48.6**	**57.6**	**84.1**	**73.2**	**87.4**	**46.0**	**59.9**	**44.6**	**28.3**	**52.9**	**55.8**	**36.4**
InternLM2 (20B)	ZSP	42.3	20.9	29.7	47.7	31.9	55.9	29.8	42.6	34.3	21.2	40.0	34.7	27.2
	FSP	55.9	27.7	32.9	45.9	38.2	65.3	31.3	46.7	37.1	25.4	43.4	37.9	29.3
	OUR	**57.8**	**53.1**	**60.5**	**83.9**	**72.3**	**91.1**	**49.7**	**59.0**	**48.2**	**31.8**	**55.9**	**63.3**	**40.5**

Table 11: The detailed **sentence** level correction F_1 score.

Datasets		**rSighans**			**ECSpell**			**Lemon**						
Subsets		Y13	Y14	Y15	Law	Med	Odw	Car	Cot	Enc	Gam	Med	New	Nov
Domain-Specific SOTAs (*Trained on in-domain gold-standard data of each dataset*)														
ReaLiSe		85.0	76.3	80.9	*48.7*	*34.4*	*53.0*	*37.4*	*42.7*	*32.9*	*16.3*	*33.8*	*35.1*	*23.2*
Domain-General SOTAs (*Trained on about 34M synthetic CSC data*)														
Finetuned BERT		64.3	51.0	57.2	66.3	59.0	69.5	53.0	64.1	46.0	35.6	52.3	57.5	36.3
Softmasked BERT		**65.6**	49.3	57.3	67.2	61.3	70.0	53.6	63.3	45.4	31.6	51.0	57.9	**38.5**
ReLM		58.6	**51.1**	**61.0**	**68.3**	**63.9**	**73.0**	**54.4**	**66.1**	**48.2**	**37.5**	**55.1**	**60.5**	37.1
LLMs (*without CSC-specific training*)														
Baichuan2 (13B)	ZSP	29.6	11.2	14.5	20.5	16.6	29.8	7.8	7.4	12.5	4.1	11.9	14.2	10.6
	FSP	51.8	29.7	34.0	54.9	52.5	51.8	14.0	35.3	23.0	9.5	29.5	39.0	26.2
	OUR	**79.1**	**66.3**	**67.3**	**88.8**	**86.7**	**93.8**	**57.5**	**64.0**	**56.5**	**39.6**	**61.7**	**66.2**	**47.9**
Qwen1.5 (14B)	ZSP	48.8	18.9	26.5	53.5	35.4	58.1	27.1	26.8	32.0	12.7	32.1	35.1	21.5
	FSP	51.0	29.5	33.2	63.3	44.4	66.9	22.7	39.8	34.7	14.3	34.9	36.5	28.4
	OUR	**75.2**	**62.8**	**66.0**	**88.6**	**84.5**	**91.6**	**52.4**	**62.9**	**49.6**	**34.3**	**54.6**	**59.5**	**42.6**
InternLM2 (20B)	ZSP	46.0	18.1	27.3	40.5	22.8	49.3	24.7	31.9	29.7	12.3	31.0	29.2	26.6
	FSP	46.8	25.5	33.4	56.7	40.0	66.3	24.5	34.2	30.4	10.4	40.9	32.9	28.9
	OUR	**76.8**	**65.5**	**67.8**	**88.9**	**83.6**	**93.8**	**54.6**	**62.0**	**53.1**	**36.7**	**57.9**	**65.9**	**45.3**

Table 12: The detailed **character** level correction F_1 score.

models show limited generalization beyond their maximum training length. We truncate inputs to a maximum length of 128 characters and concatenate the remaining characters to the output.

LLM baselines The outputs of LLMs sometimes fail to align with evaluation, primarily due to their inadequate instruction-following capability. To address this, we apply specific rules for post-processing: 1) *Redundant Phrases:* We remove redundant phrases such as "修改后的句子是：" (*The corrected sentence is:*), identified through common patterns input in the model output; 2) *Redundant Punctuation:* Many sentences in the dataset lack terminal periods, yet some models inappropriately add them. To prevent incorrect evaluations due to this discrepancy, we remove any

Datasets		rSighans			ECSpell			Lemon						
Subsets		Y13	Y14	Y15	Law	Med	Odw	Car	Cot	Enc	Gam	Med	New	Nov
Domain-Specific SOTAs (*Trained on in-domain gold-standard data of each dataset*)														
ReaLiSe		13.0	9.6	7.7	*10.6*	*18.6*	*11.8*	*20.9*	*13.4*	*20.8*	*22.5*	*16.5*	*16.7*	*22.6*
Liu et al. (2023)		–	–	–	7.4	6.5	2.2	–	–	–	–	–	–	–
Domain-General SOTAs (*Trained on about 34M synthetic CSC data*)														
Finetuned BERT		21.7	16.5	12.5	**4.9**	11.3	**2.9**	12.3	8.3	13.9	22.5	8.3	9.4	17.3
Softmasked BERT		13.0	17.6	14.5	6.1	11.7	5.0	12.4	7.1	14.8	**20.4**	9.6	10.6	**16.6**
ReLM		**4.4**	**15.0**	**9.5**	7.8	**11.0**	7.1	**12.1**	**5.6**	**12.6**	20.8	**5.7**	**8.4**	17.5
LLMs (*without CSC-specific training*)														
Baichuan2 (13B)	ZSP	34.8	58.3	54.4	26.9	43.1	21.0	40.6	54.2	35.9	41.6	35.4	41.1	37.6
	FSP	21.7	19.4	23.2	7.8	**9.1**	**0.4**	8.3	7.4	10.2	20.0	4.6	8.3	**7.7**
	OUR	**8.7**	**14.1**	**8.3**	**4.5**	9.9	**0.4**	**5.9**	**6.9**	**8.9**	**19.2**	**3.9**	**5.7**	13.0
Qwen1.5 (14B)	ZSP	34.8	54.4	34.2	5.7	35.4	2.1	18.5	15.8	13.5	18.4	11.8	14.0	20.7
	FSP	**15.9**	30.9	31.7	5.3	**11.6**	**0.8**	**8.9**	12.7	**10.1**	**14.7**	9.5	**7.8**	**5.5**
	OUR	21.7	**19.6**	**10.2**	**4.9**	11.7	2.9	11.2	**6.3**	14.8	29.4	**5.4**	10.1	21.2
InternLM2 (20B)	ZSP	65.2	58.0	48.8	26.5	50.7	17.7	28.8	23.7	30.0	30.6	23.0	34.0	24.2
	FSP	21.7	39.8	33.6	13.9	30.7	2.5	18.2	12.3	18.1	23.7	10.0	22.4	16.1
	OUR	**13.0**	**16.5**	**8.3**	**2.5**	**12.4**	**0.4**	**8.5**	**6.9**	**12.2**	**22.5**	**3.7**	**6.1**	**15.1**

Table 13: The detailed sentence level false positive rate.

Input	商务部前头，11月底完成
Reference	商务部牵头，11月底完成
ReLM	商务部牵头，11月底完成
BC2 13B ZSP	商务部前面，11月底完成
BC2 13B FSP	商务部日前，11月底完成
BC2 13B OUR	商务部牵头，11月底完成
Input	虎珀酸索莉那新片主要功能是什么
Reference	琥珀酸索利那新片主要功能是什么
ReLM	琥珀酸索莉那新片主要功能是什么
BC2 13B ZSP	琥珀酸索利那新片主要功能是什么
BC2 13B FSP	虎珀酸索莉那新片主要功能是什么
BC2 13B OUR	琥珀酸索利那新片主要功能是什么

Table 14: Qualitative examples of our approach and the baselines. Corrections marked in "Blue" are correct or suggested by the reference, while those in "Red" are incorrect.

added terminal period if the original sentence did not have one.

C More Results

C.1 Detailed Results

Due to the space limitation, we only present the average results of each dataset in the main text. The detailed results of each dataset are shown in Table 11, Table 12, and Table 13.

C.2 Qualitative Examples

We provide two qualitative examples to illustrate the performance of our approach in Table 14.

In the first case ("*Led by the Ministry of Commerce, to be completed by the end of November*"), the word "牵头" (*qiāntóu, led by*) is misspelled as "前头" (*qiántóu, front*) in the input sentence. Both the ZSP and FSP baselines mistakenly put their attention on the character "前" (*front*) and incorrectly correct "前头" to "日前" (*a few days ago*) and "前面" (*front*), respectively. Such corrections are not only implausible but also linguistically awkward. In contrast, the domain-general model ReLM and our approach successfully correct the misspelling.

In the second case ("*What are the main functions of Solifenacin Succinate Tablets*"), the name of the drug "琥珀酸索利那新片" (*Solifenacin Succinate Tablets*) is misspelled. To correct the misspelling, the knowledge of the medical domain is required. In this case, the ReLM model fails to correct the misspelling, while the zero-shot prompting baseline and our approach successfully correct it. It is worth noting that the few-shot prompting baseline also fails to correct the misspelling, which indicates that the inclusion of inappropriate examples may lead to worse performance.

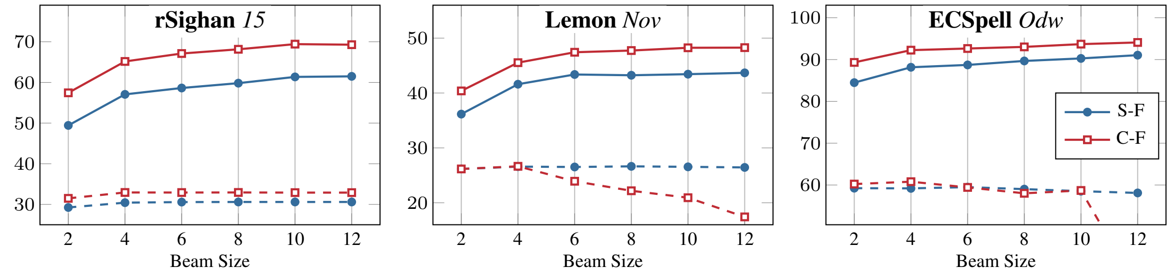


Figure 5: The scores of Baichuan2 7B with different beam sizes. The solid lines represent the results of our approach, and the dashed lines represent the results of the few-shot baseline. We can observe that larger beam sizes may lead to worse C-F scores in few-shot settings.

	rSighan *15*		**Lemon** *Nov*		**ECSpell** *Odw*	
	Dev	True	Dev	True	Dev	True
Distortion Model: $\log p_{\texttt{DM}}$						
Idt.	-0.04	-0.03	-0.04	-0.02	-0.04	-0.02
Sa.P.	-3.75	-4.00	-3.75	-4.66	-3.75	-4.17
Si.P.	-4.85	-5.02	-4.85	-5.45	-4.85	-5.87
Si.S.	-5.40	-8.63	-5.40	-8.04	-5.40	-6.66
S-F$^{\uparrow}$	59.8	**+0.9**	43.2	0.0	89.7	-0.8
C-F$^{\uparrow}$	68.2	**+1.4**	47.7	**+0.2**	93.0	-0.3
FPR$^{\downarrow}$	8.1	0.0	13.6	+0.3	1.3	0.0

Table 15: The impact of distortion model on the performance of Baichuan2 7B. "True" denotes that the distortion model is derived from the **true** distortion distribution of each dataset. " Dev" represents the distortion model from the Pseudo-Dev.

D More Discussions

D.1 Influence of Beam Size

During searching the most likely correction sequence, the beam search algorithm is used to avoid the exponential growth of the search space and the local minimum caused by greedy search. Knowing the impact of the beam size on the performance helps researchers to choose a proper beam size to balance the trade-off between the performance and the computational cost. The results are shown in Figure 5. Though the larger beam size consistently leads to better performance, the improvement becomes marginal when the beam size is larger than 6.

D.2 Effectiveness of the Estimated Distortion Model

The distortion model is a key component in our approach. In this work, we utilize a minimal distortion model and directly estimate the distortion probabilities from the statistics of the Pseudo-Dev dataset. Obviously, this estimation will be different from the true probabilities.

To verify the effectiveness of the estimated distortion model, we conduct experiments comparing the estimated distortion model with the true distortion model. The results are presented in Table 15. The upper part of the table shows the difference between the estimated distortion model and the true distortion model. We can see that the estimated one is quite close to the true one, except for the Similar Shape distortion type. The lower part shows that the difference between the performance is marginal, indicating that the estimated distortion model is sufficient for our approach to achieve a good performance, and has good generalization ability across different datasets.

System		**Inference Speed** (ms)	
		per **Sent.**	*per* **Char.**
ReLM		14.4	0.4
Baichuan2 13B	ZSP	899.8	22.2
	FSP	1,057.4	26.1
	OUR	1,541.0	38.0

Table 16: The inference speed of different models.

D.3 Inference Speed

We conducted a brief runtime analysis to evaluate the inference speed of our approach. The analysis was performed using a single NVIDIA A100 40GB GPU with an Intel Xeon Gold 6248R (3.00GHz) CPU. The batch size was set to 1 for all models, and other hyperparameters were set to the same values as in the main experiments.

The average inference speed of each model on the ECSpell-Odw dataset is shown in Table 16. Due to the large model size and the autoregressive decoding process, LLMs are significantly slower than the BERT-based ReLM model. Compared to the ZSP and FSP baselines, our approach is slower (1.71× and 1.45×, respectively), primarily due to our immature implementation of the distortion model, which can be further optimized to improve inference speed.

D.4 Does Our Approach Work Well on Simpler LMs?

Though we mainly focus on the performance of our approach on LLMs, the language model part of Equation 1 can be replaced by other simpler models, such as n-gram, masked language model, or small-scale causal language model. In this subsection, we investigate the performance of our approach on simpler LMs.

Though our primary focus is on the performance of our approach on LLMs, the language model term of Equation 1 can be substituted with simpler models, such as n-gram models, masked language models, or small-scale causal language models. In this subsection, we investigate the performance of our approach using these simpler language models.

The LMs we investigate include:

- **n-gram LM**: `KLM`,[9] a 5-gram language model trained on the Chinese Gigaword corpus.
- **Masked LM**: `BERT`,[10] a bidirectional language model pre-trained using the mask filling task and next sentence prediction task.
- **Small causal LM**: `GPT2`,[11] a small-scale causal language model (about 102M parameters) trained on the CLUECorpusSmall dataset (about 5B characters).

The results are shown in Table 17. From these results, we can see that our approach also works with simpler LMs. In the ECSpell-Odw dataset, our approach enables simpler language models (LMs) to achieve sentence- and character-level correction F1 scores higher than 50% and 60%, respectively. However, the performance of our approach on simpler LMs still lags significantly behind that of the large language models (LLMs), highlighting the importance of the scale of pre-training data and model size.

D.5 Impact of the Pre-training Data

There are two main factors that differentiate LLMs from simpler LMs: the scale of pre-training data and the model size. The impact of model size on the performance of LLMs has been discussed in §5.1. In this subsection, we aim to investigate the impact of pre-training data on the performance of our approach.

We compare `Qwen1.5`, a recent LLM family, with `GPT2`, which also has a causal LM (decoder-only) architecture. The `GPT2` model family partially overlaps in model size with the `Qwen1.5` model family, but it was trained on a much smaller dataset, CLUECorpusSmall. The CLUECorpusSmall dataset contains only about 5 billion characters and has limited diversity in text sources, including only news, Wikipedia, forums, and comments.

As shown in Table 18, when the model sizes are similar, the `Qwen1.5` model family outperforms the `GPT2` model family on all three datasets. The largest performance gap is observed on the Lemon-Nov dataset, where a smaller 463M `Qwen1.5` model even outperforms a larger 1.5B `GPT2` model by 7.1% in the sentence-level correction F_1 score. This is because the Lemon-Nov dataset contains texts from the novel domain, which is not included in the CLUECorpusSmall dataset. These results indicate that the scale and diversity of the pre-training data are crucial for the performance of our approach.

D.6 Comparison to the Supervised Fine-tuning Method

In this subsection, we compare our approach with the supervised fine-tuning method.

However, we did not fine-tune the LLMs ourselves, as fine-tuning an LLM on the 34M synthetic CSC data would be extremely time-consuming and computationally expensive. Additionally, the supervised fine-tuning method typically requires careful hyperparameter tuning to achieve the best performance, further increasing the computational cost.

Instead, we leverage the findings from Li et al. (2023a), who fine-tuned the `Baichuan2 7B` and `GPT2` models on the ECSpell dataset.

The results are shown in Table 19. Compared to the ReLM model, the supervised fine-tuning method is less effective in improving the performance of causal LMs like `GPT2` and recent LLMs such as `Baichuan2`. In some cases, our training-free approach even outperforms the supervised fine-tuning counterpart.

This phenomenon can be attributed to the characteristics of the ECSpell dataset, which, as pointed out by Wu et al. (2023), contains a high proportion (more than 70%) of error-correction pairs that never appeared in the training data. The supervised fine-tuning method is not effective in handling these unseen error-correction pairs, whereas our approach can still correct them.

[9] `shibing624/chinese-kenlm-klm`

[10] `bert-base-chinese`

[11] `uer/gpt2-chinese-cluecorpussmall`

System	rSighan *15*			Lemon *Nov*			ECSpell *Odw*		
	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓
KLM	29.3	38.9	33.8	5.8	9.4	65.8	58.3	65.3	23.5
BERT 110M	31.3	34.0	0.2	13.3	12.5	0.6	59.1	63.6	0.0
GPT2 102M	55.0	64.7	8.1	26.1	30.8	28.4	78.6	85.0	5.4
Baichuan2 13B	59.6	67.3	8.3	43.5	47.9	13.0	92.0	93.8	0.4
Qwen1.5 14B	57.6	66.0	10.2	36.4	42.6	21.2	87.4	91.6	2.9
InternLM2 20B	60.5	67.8	8.3	40.5	45.3	15.1	91.1	93.8	0.4

Table 17: Results of applying our approach to different models.

System	Data Amount	rSighan *15*			Lemon *Nov*			ECSpell *Odw*		
		S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓	S-F↑	C-F↑	FPR↓
GPT2 1.5B	Small	56.6	64.4	10.4	26.1	31.8	31.4	82.8	85.8	5.5
Qwen1.5 463M	Large	56.3	63.5	10.0	33.2	40.2	22.2	84.7	89.9	3.8
Qwen1.5 1.8B		58.3	65.3	10.3	35.6	42.3	19.9	90.3	92.8	1.7

Table 18: A brief comparison of the performance of LLMs of different sizes and pre-training data amounts on three datasets.

Model	Method	Law	Med	Odw
ReLM	34M-ft	60.4	50.9	66.5
	Id-ft	91.2	82.4	83.6
GPT2 110M	Id-ft	71.2	35.6	53.8
	OUR	66.4	60.0	78.6
Baichuan2 7B	Id-ft	86.0	73.2	82.6
	OUR	82.1	79.7	89.7

Table 19: The sentence-level correction F_1 scores of models supervised fine-tuned on in-domain training data and our approach on ECSpell datasets. Id-ft denotes the model fine-tuned on the training data of ECSpell.

D.7 How to Introduce New Knowledge into Our Approach?

The LLM part of our approach offers a straightforward way to incorporate new knowledge **without the need for further training**, by **adding some text that describes the new knowledge as an input prefix**.

Given the new knowledge $\boldsymbol{k}$, Equation 1 can be adjusted from $p(\boldsymbol{x}, \boldsymbol{y})$ to $p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{k})$. We then have:

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{k}) &= p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{k})\, p(\boldsymbol{y} \mid \boldsymbol{k}) \\ &\approx p_{\mathsf{DM}}(\boldsymbol{x} \mid \boldsymbol{y})\, p_{\mathsf{LLM}}(\boldsymbol{y} \mid \boldsymbol{k}), \end{aligned} \tag{9}$$

where, by assuming $\boldsymbol{x}$ and $\boldsymbol{k}$ are conditionally independent given $\boldsymbol{y}$, we approximate $p(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{k})$ as $p_{\mathsf{DM}}(\boldsymbol{x} \mid \boldsymbol{y})$. The second term, $p_{\mathsf{LLM}}(\boldsymbol{y} \mid \boldsymbol{k})$, can be calculated by the LLM using the input prefix $\boldsymbol{k}$.

To illustrate this point, we conducted a simple experiment introducing domain and text format information as new knowledge into our approach. We chose the **MCSCSet** dataset for this experiment, as the sentences in this dataset share a common characteristic: they are *questions from patients*. We can introduce this knowledge into the LLM by adding a simple input prefix $\boldsymbol{k}$ = "患者提问：" ("*A patient asks:*").

The results in Table 20 demonstrate that introducing new knowledge into the LLM by merely modifying the input prefix can significantly improve the model's performance on the CSC task. Notably, this simple method also works well on the BERT-based baselines, yielding improvements of 0.7% to 1.2% in the sentence-level F_1 score and 1.7% to 2.1% in the character-level F_1 score, although the improvements are not as significant as those observed with the LLMs.

We provide a real case from the MCSCSet dataset to explain why this method works.

Consider the sentence "未挨前兆" (wèi āi qián zhào, "*without being near any prior warnings*"), which should be corrected to "胃癌前兆" (wèi ái qián zhào, "*early symptoms of stomach cancer*") in the medical domain. This sentence contains only four characters, insufficient to provide enough context for accurate spelling correction, even for humans.

CSC models often fail to correct this sentence or suggest incorrect corrections, such as "未提前兆" (wèi tí qián zhào, "*did not provide prior warnings*") or "未按前兆" (wèi àn qián zhào, "*not

System		MCSCSet		
		S-F$^{\uparrow}$	C-F$^{\uparrow}$	FPR$^{\downarrow}$
`Finetuned BERT`	`ORI`	35.3	48.5	7.5
	`w/` $\boldsymbol{k}$	+0.7	+1.7	+0.1
`Softmasked BERT`	`ORI`	35.3	48.5	8.1
	`w/` $\boldsymbol{k}$	+1.2	+2.1	-0.5
`ReLM`	`ORI`	37.8	50.2	6.8
	`w/` $\boldsymbol{k}$	+0.9	+1.9	-0.2
`Baichuan2 13B`	`OUR`	66.0	76.9	1.7
	`w/` $\boldsymbol{k}$	+5.1	+5.4	-0.2
`Qwen1.5 14B`	`OUR`	61.1	72.6	3.1
	`w/` $\boldsymbol{k}$	+9.1	+8.8	-1.0
`InternLM2 20B`	`OUR`	63.2	72.9	2.6
	`w/` $\boldsymbol{k}$	+4.8	+5.4	-0.0

Table 20: The results of introducing new knowledge by adding a prefix $\boldsymbol{k}$ to the input. “`ORI`” denotes the original input without any prefix.

according to the prior warnings”). However, if we add the prefix “患者提问：” (“*A patient asks:*”), which provides the knowledge that the sentence is a patient’s question about a medical condition, the model can make the correct correction to “胃癌前兆”.