A Training-free LLM-based Approach to General Chinese Character Error Correction

Houquan Zhou[1], **Bo Zhang**[2], **Zhenghua Li**[1✉]
Ming Yan[2], **Min Zhang**[1]
[1]School of Computer Science and Technology, Soochow University, China
hqzhou@stu.suda.edu.cn, {zhli13,minzhang}@suda.edu.cn
[2]Alibaba Group, China
{klayzhang.zb,ym119608}@alibaba-inc.com

Abstract

Chinese spelling correction (CSC) is a crucial task that aims to correct character errors in Chinese text. While conventional CSC focuses on character substitution errors caused by mistyping, two other common types of character errors, missing and redundant characters, have received less attention. These errors are often excluded from CSC datasets during the annotation process or ignored during evaluation, even when they have been annotated. This issue limits the practicality of the CSC task. To address this issue, we introduce the task of General Chinese Character Error Correction (C2EC), which focuses on all three types of character errors. We construct a high-quality C2EC benchmark by combining and manually verifying data from CCTC and Lemon datasets. We extend the training-free prompt-free CSC method to C2EC by using Levenshtein distance for handling length changes and leveraging an additional prompt-based large language model (LLM) to improve performance. Experiments show that our method enables a 14B-parameter LLM to be on par with models nearly 50 times larger on both conventional CSC and C2EC tasks, without any fine-tuning.

1 Introduction

Given an input sentence $\boldsymbol{x} = x_1, x_2 \cdots x_n$, the task of conventional Chinese spelling correction (CSC) aims to produce a new sentence of the same length, denoted as $\boldsymbol{y} = y_1, y_2 \cdots y_n$, where each misspelled character (*e.g.*, x_i) is replaced with a correct one (*e.g.*, y_i). Spelling errors in text can cause misunderstandings, damage authenticity, and even lead to unnecessary financial losses, making automatic correction a crucial task in Chinese Natural Language Processing (NLP) (Wu et al., 2023; Zhou et al., 2024; Li et al., 2024; Dong et al., 2024; Liu et al., 2024a).

General Chinese Character Error Correction

Misspelling Character (*Conventional Chinese Spelling Correction*)
我这学期选修了[羽矛球→羽毛球]课。
I have taken [badninton→badminton] as an elective this semester.

Missing Character (*New*)
我这学期选修了[羽球→羽毛球]课。
I have taken [badinton→badminton] as an elective this semester.

Redundant Character (*New*)
我这学期选修了[羽矛毛球→羽毛球]课。
I have taken [badmintton→badminton] as an elective this semester.

Figure 1: Scope of the general Chinese character error correction. The correct sentence should be "我这学期选修了羽毛球课。" (*I have taken badminton as an elective this semester.*).

Spelling errors mainly arise from two sources: The first source is typing mistakes. Chinese characters often have multiple visually or phonetically similar variants, making it easy to select an incorrect one when using input methods (Hu et al., 2024). The second source is automatic text conversion errors. When using automatic speech recognition (ASR) or optical character recognition (OCR), systems may also introduce incorrect characters during the conversion process (Wang et al., 2018).

Beyond spelling errors, character errors involving missing and redundant characters are also common (He et al., 2023). These errors can be caused by the same reasons as spelling errors. For instance, when typing "曲安" (*qū ān*) in "醋酸曲安奈德" (*Triamcinolone Acetonide*), the input method might consider "*qū ān*" as a single character "圈" (*quān*), resulting in a misspelled and missing character error.[1] Additionally, repetitive sentence editing, such as when revising messages, social media posts, or emails, can easily introduce missing and redundant errors. Consider this example: "我参加的

✉ Zhenghua Li is the corresponding author.

[1]A real case from the Lemon *Mec* subset.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13827–13852
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

项目是羽球。" (*The project I participated in is badmton*), where "毛" was omitted from "羽毛球" (*badminton*). The missing "毛" likely occurred during editing: when correcting a mistyped "求" to "球", the user accidentally deleted two characters before retyping "球". Such errors often go unnoticed without thorough proofreading. For clarity, we refer to misspelling, missing, and redundant errors as **General Chinese Character Errors** (C2E).

Recent Chinese text correction competitions, CTC 2021 (Zhao et al., 2022), Midu-CTC[2], and Kingsoft-CTC[3], have designed error distributions to reflect real-world scenarios. In these competitions, C2E constitute 79.4% to 87.3% of errors (Table 1). This indicates that C2E are more common than complex errors like grammatical mistakes, logical inconsistencies, or ambiguity. Compared to conventional CSC, addressing C2E is more practical as it covers a broader range of common errors.

Thus, we believe that **General Chinese Character Error Correction** (C2EC) deserves more attention from the Chinese text correction community. While He et al. (2023) previously studied C2EC, they created synthetic ECMR-2023 by placing random errors into correct sentences. There is a lack of a dataset specifically focusing on C2E with real-world errors. To fill this gap, we build a new C2EC dataset using two existing datasets containing real-world errors: CCTC (Wang et al., 2022) and Lemon (Wu et al., 2023). We carefully verified the data to ensure data quality and annotation consistency, resulting in 1,995 sentences for development and 5,711 for testing.

Recent work has shown the power of large language models (LLMs) for CSC (Dong et al., 2024; Li et al., 2024; Zhou et al., 2024). Notably, by combining an LLM, which evaluates fluency, with a Hamming distance to prevent over-correction, the training-free prompt-free framework (`TfPf`) (Zhou et al., 2024) achieves strong results without any training. We extend this approach to C2EC by using Levenshtein distance to handle missing and redundant errors, and incorporating prompt-based probability scoring for better performance.

Experiments show that our approach achieves large improvements over the `TfPf` baseline on both conventional CSC and C2EC datasets, and even outperforms the supervised fine-tuned counterparts on various domains. It is also worth mentioning that our approach enables a 14B parameter model to achieve competitive performance with models nearly 50 times larger without any training.

Our code is available at `https://github.com/Jacob-Zhou/simple-csc/tree/v2.0.0`.

[2] `https://aistudio.baidu.com/competition/detail/404/0/introduction`

[3] `https://datastudio.wps.cn/matchcenter/competition/1/introduction`

2 The C2EC Task

2.1 Task Definition

Given an input sentence $\boldsymbol{x} = x_1, x_2 \cdots x_n$, the task of C2EC aims to correct character errors and produce a corrected sentence $\boldsymbol{y} = y_1, y_2 \cdots y_m$. Unlike conventional CSC, where input and output lengths must match, C2EC allows $\boldsymbol{y}$ to have a different length from $\boldsymbol{x}$. In addition to character substitutions, C2EC addresses two additional error types: *a)* **Missing Character**: where characters are missing from the input, e.g., "羽球" missing "毛" from "羽毛球" (*badminton*). *b)* **Redundant Character**: where unexpected extra characters appear in the input, e.g., the extra "矛" in "羽矛毛球".

2.2 Construction of C2EC Dataset

Rather than using rule-based synthesis like He et al. (2023), we construct our dataset from existing datasets to better reflect real-world error patterns.

Data Selection and Division We build the C2EC dataset from two high-quality sources: *a)* **CCTC** (Wang et al., 2022): A comprehensive dataset of ~25,000 sentences covering both character errors and complex errors from diverse sources. While this dataset includes C2E errors, it was not specifically designed to focus on C2EC. *b)* **Lemon** (Wu et al., 2023): A CSC dataset containing ~500 sentences with missing and redundant character errors. Although it contains sentences with C2E errors, previous works excluded these when evaluating CSC methods (Wu et al., 2023; Liu et al., 2024b; Zhou et al., 2024; Li et al., 2024). We use CCTC's training set for development and combine the test sets from both CCTC and Lemon for testing.

Data Resampling To achieve a more balanced and natural error distribution, we perform two resampling steps: *a)* We balance the ratio of correct to incorrect sentences to 1:1, reducing the original 91% correct sentence bias in CCTC. *b)* We adjust Lemon's error type distribution to match CCTC's (~75% substitution, ~25% missing/redundant), addressing the bias of Lemon toward substitution errors (95%). Throughout resampling, we maintain

Datasets	# Sent	% Err. Sent	Avg. Len.	# Char. Error SUB	RED	MIS	*All*	# Other Error
CTC21 *Qua*	969	49.5	48.9	280 (*65.6*)	59 (*13.8*)	88 (*20.6*)	427 [*79.4*]	111 [*20.6*]
MiduCTC *Val*	1,014	50.7	57.3	389 (*82.2*)	35 (*7.4*)	49 (*10.4*)	473 [*85.7*]	79 [*14.3*]
KingsoftCTC *Labeled*	2,000	100.0	79.4	2,538 (*75.1*)	607 (*18.0*)	236 (*7.0*)	3,381 [*87.3*]	493 [*12.7*]
CCTC *Test*	13,037	9.3	41.1	933 (*74.9*)	146 (*11.7*)	166 (*13.3*)	1,245 [*92.6*]	100 [*7.4*]
Lemon *All*	22,252	50.0	35.4	12,094 (*94.8*)	369 (*2.9*)	288 (*2.3*)	12,751	–
ECMR-2023 *Test*	1,000	N/A	N/A	3,105 (*81.6*)	456 (*12.0*)	246 (*6.5*)	3,807	–
C2EC *Dev*	1,995	50.1	51.9	818 (*74.5*)	148 (*13.5*)	132 (*12.0*)	1,098	–
C2EC *Test*	5,711	49.9	41.9	2,397 (*72.6*)	462 (*14.0*)	442 (*13.4*)	3,301	–

Table 1: The statistics of the datasets. Numbers in the parentheses "()" are the percentages of the total number of character errors. Numbers in the brackets "[]" are the percentages of the total number of all errors.

balanced distributions across domains and correct-incorrect sentence pairs.

Quality Control To maintain focus on character-level errors and keep the dataset clean, we apply two data cleaning processes: *a)* Automatically remove sentences with complex errors, specifically those errors that involve continuous insertions or deletions (this removed 91 and 90 sentences from the development and test sets, respectively). *b)* Manually verify sentences by native Chinese speakers to remove sentences that: *i.* Have incorrect annotations; *ii.* Have complex grammatical errors; *iii.* Have multiple reasonable corrections; *iv.* Are ambiguous or difficult to understand. Four native Chinese speakers familiar with the CSC task performed the manual verification. Each sentence was reviewed by at least two annotators, with a third resolving any disagreements. The inter-annotator agreement reached 97.08%. This process removed 111 test set and 31 development set sentences.[4]

2.3 Data Statistics

The final dataset, as shown in Table 1, contains 1,995 development and 5,711 test sentences, with approximately half being error-free. The development set contains 1,098 errors (~1.1 per erroneous sentence), while the test set has 3,301 errors (~1.2 per erroneous sentence). In the test set, 72.6% of errors are misspellings, 14.0% are redundant characters, and 13.4% are missing characters.

3 The Baseline TfPf Approach

The training-free prompt-free framework (`TfPf`) (Zhou et al., 2024) combines a large language model with a distance metric:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \log p_{\texttt{LLM}}(\boldsymbol{y}) - \texttt{Dist}(\boldsymbol{x}, \boldsymbol{y}) \quad (1)$$

The first term $p_{\texttt{LLM}}(\boldsymbol{y})$ represents a large language model that models the probability of the correct sentence $\boldsymbol{y}$, ensuring the correction is fluent from a language perspective. The second term $\texttt{Dist}(\boldsymbol{x}, \boldsymbol{y})$ measures the Hamming distance[5] between $\boldsymbol{x}$ and $\boldsymbol{y}$, preventing the model from making too many changes or inserting/deleting characters in the input sentence to achieve fluency.

4 Our Training-free Approach for C2EC

We extend the `TfPf` framework with the following equation:

$$\begin{aligned} s(\boldsymbol{x}, \boldsymbol{y}) = & \log p_{\texttt{LLM}}(\boldsymbol{y} \mid \texttt{Prompt}(\boldsymbol{x})) \\ & + \log p_{\texttt{LLM}}(\boldsymbol{y}) - \texttt{Dist}_{\texttt{L}}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \quad (2)$$

where $\texttt{Prompt}(\cdot)$ is the prompt template.

Our approach makes three key extensions to the `TfPf` framework: *a)* We allow the input sentence $\boldsymbol{x}$ to have a different length from the output sentence $\boldsymbol{y}$. *b)* We replace the original Hamming distance metric with Levenshtein distance to handle varying input and output lengths. We use $\texttt{Dist}_{\texttt{L}}$ to distinguish it from the original distance metric $\texttt{Dist}$. *c)* We incorporate an additional prompt-based probability $p_{\texttt{LLM}}(\boldsymbol{y} \mid \texttt{Prompt}(\boldsymbol{x}))$ in the score function to improve performance.

4.1 Incremental Weighted Levenshtein Distance for Generation

Following `TfPf`, we use a weighted distance metric to measure the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. We classify each edit operation into different types and

[4]Several examples of discarded sentences are provided in the appendix A.2.

[5]The original `TfPf` paper refers to this part as the distortion model. However, from a mathematical perspective, the distortion model is equivalent to minus the weighted Hamming distance between $\boldsymbol{x}$ and $\boldsymbol{y}$. Details are in Appendix B.1.

Edit Type	Example	Weight
Keep	机器 (*jī qì*)	0.04
Substitute		
Same Pinyin	基器 (*jī qì*)	3.75
Similar Pinyin	七器 (*qī qì*)	4.85
Similar Shape	仉器 (*zhǎng qì*)	5.40
Other Similar	金器 (*jīn qì*)	8.91
Insert	机器 → 机器人	8.50
Delete	机器 → 器	9.00

Table 2: Examples of the different edit types of the corrected token "机器" (*jī*). We adopt the weight of substitution from Zhou et al. (2024).

```
Minimal Prompt

纠正下面句子中的错别字以及多字少字错误，并给出修改后的句子。\n
输入：{INPUT: x}\n
输出：
----------------------------------------------
Correct the typos and redundant or missing characters in the following sentence, and provide the sentence after correction.\n
Input: {INPUT: x}\n
Output:
```

Figure 2: A minimal prompt template used in our method.

assign a specific weight to each type. Table 2 shows the weights used in our work. For operations already defined in `TfPf`, we adopt their original weights[6]. For the two new operation types, `Insert` and `Delete`, we set weights to 8.5 and 9.0, respectively, after grid search on development sets.

Large language models generate output incrementally, while Levenshtein distance requires solving a dynamic programming problem over the entire sentence.

To mitigate this incompatibility, we introduce an incremental Levenshtein distance. For a partial output candidate $\boldsymbol{y}_{\leqslant j}$, we define the partial distance as the Levenshtein distance between $\boldsymbol{y}_{\leqslant j}$ and a corresponding prefix of the input sentence $\boldsymbol{x}_{\leqslant b}$. The incremental Levenshtein distance after generating a new character y_j is:

$$\begin{aligned}&\Delta_{\texttt{Dist}_\texttt{L}}(\boldsymbol{x}, a, b, \boldsymbol{y}_{<j}, y_j)\\&\quad = \texttt{Dist}_\texttt{L}(\boldsymbol{x}_{\leqslant b}, \boldsymbol{y}_{<j} \circ y_j) - \texttt{Dist}_\texttt{L}(\boldsymbol{x}_{\leqslant a}, \boldsymbol{y}_{<j})\end{aligned} \tag{3}$$

where $\circ$ denotes concatenation, and a and b are the end indices of the input prefix before and after generating y_j, respectively.

4.2 Extra Use of Prompt-based LLM

A Reinforcement Learning Perspective The use of two LLMs in Equation 2 can be understood from a reinforcement learning perspective. In this view, the `TfPf` framework serves as a reward function that guides the generation process. While the prompt-based LLM acts as a reference model that produces correction probabilities based on the prompt, the pure LLM evaluates the fluency of generated sequences. Details on this perspective are provided in Appendix C.

Selection of LLM To save GPU memory, we use the same model for both the prompt-based LLM and language model functions. When GPU memory is sufficient, different models could be used for each function.

We use the base version of the LLM for both the prompt-based LLM and the language model, as our experiments show that it is more robust than instruction-tuned versions across different prompt templates and achieves better overall performance.

Prompt Template We designed two prompt templates: a minimal prompt (`Minimal`) based on Li et al. (2024) and a more sophisticated prompt (`Detailed`) based on Li et al. (2023); Zhou et al. (2024). Figure 2 shows the minimal prompt, which contains only essential task description and input sentence. The detailed prompt, shown in Figure 6 in Appendix 4.2, additionally includes task-specific system prompts, format requirements, and notes.

Our experiments show that the base version of LLM performs better with the minimal prompt, while instruction-tuned LLM works better with the detailed prompt. Based on overall performance, we adopt the minimal prompt in this work.

4.3 Token-based Generation and Beam Search

Large language models use multi-character tokenization for Chinese. A token $\boldsymbol{t}_k$ may contain multiple characters. For clarity, we define a partial output candidate at generation step k as $\boldsymbol{y}_{\leqslant j} = y_1 \cdots y_j = \boldsymbol{t}_1 \cdots \boldsymbol{t}_k$.

At each generation step k, we calculate the token

[6] https://github.com/Jacob-Zhou/simple-csc/blob/main/configs/default_config.yaml

score $\boldsymbol{t}_k$ as:

$$\begin{aligned} s(\boldsymbol{x}, \boldsymbol{t}_1 \ldots \boldsymbol{t}_k, b) &= s(\boldsymbol{x}, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, a) \\ &+ \log p_{\texttt{LLM}}(\boldsymbol{t}_k \mid \texttt{Prompt}(\boldsymbol{x}) \circ \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}) \\ &+ \log p_{\texttt{LLM}}(\boldsymbol{t}_k \mid \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}) \\ &- \Delta_{\texttt{Dist}_\texttt{L}}(\boldsymbol{x}, a, b, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, \boldsymbol{t}_k) \end{aligned}$$

where $\Delta_{\texttt{Dist}_\texttt{L}}(\boldsymbol{x}, a, b, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, \boldsymbol{t}_k)$ is the incremental Levenshtein distance after generating multiple characters.

We use beam search to find the final output. That is, at each generation step k, we expand the candidates with all possible tokens $\boldsymbol{t}_k$ and end indices b and only keep the top K candidates with the highest scores. Following TfPf, the beam size is set to 8.

4.4 The Full Model

Following TfPf, we also adopt their length and faithfulness rewards in the score function:

$$\begin{aligned} s(\boldsymbol{x}; \boldsymbol{t}_1 \ldots \boldsymbol{t}_k; b) &= s(\boldsymbol{x}, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, a) \\ &+ \log p_{\texttt{LLM}}(\boldsymbol{t}_k \mid \texttt{Prompt}(\boldsymbol{x}) \circ \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}) \\ &+ \log p_{\texttt{LLM}}(\boldsymbol{t}_k \mid \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}) \\ &+ \lambda \left(\begin{array}{c} -\Delta_{\texttt{Dist}_\texttt{L}}(\boldsymbol{x}, a, b, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, \boldsymbol{t}_k) \\ + \\ \alpha\left(\texttt{len}(\boldsymbol{t}_k) - 1\right) \end{array} \right) \end{aligned} \tag{4}$$

where $\lambda = (1 + H_{\texttt{LLM}}(\cdot))$ is the faithfulness reward that dynamically increases the distance impact when the language model is uncertain, and $\alpha(\texttt{len}(\boldsymbol{t}_k) - 1)$ is the length reward encouraging longer token selection.

5 Experimental Setup

5.1 Datasets

We evaluate our approach on both conventional CSC datasets and our newly constructed dataset, **C2EC**. For conventional CSC, following Li et al. (2024), we use two representative datasets: **CSCD-NS** (Hu et al., 2024) and **Lemon** (Wu et al., 2023). Both datasets contain text written by native speakers. CSCD-NS focuses on general domain performance, while Lemon evaluates zero-shot cross-domain capabilities. The statistics of the datasets are shown in Table 7. As we are unable to access the ECMR-2023 dataset (He et al., 2023), we do not include it in our experiments.

5.2 Evaluation Metrics

Following previous works (Li et al., 2024; Zhou et al., 2024), we use character-level correction F_1 as our main evaluation metric. Since sentence-level metrics are widely used in previous works, we also report them in Appendix E.1. Details of the metrics can be found in Appendix D.4.

5.3 Baselines

We compare our approach against three training-free baselines: *a)* **In-context Learning** (ICL): This method prompts LLMs with 10 exemplars (5 erroneous, 5 correct sentences) randomly selected and shuffled for each input. During inference, we use beam search with the same beam size as our approach;[7] *b)* **Training-free Prompt-free Method** (TfPf, Zhou et al. (2024)); and *c)* **ICL with Reranking** (ICL-RR): This hybrid method first generates K candidates using ICL, then reranks them using an extended TfPf model that supports insertion and deletion operations.

For reference, we also include ICL results from leading LLMs accessed via API:[8] GPT4o-mini,[9] GPT4o (Hurst et al., 2024),[10] and the 671B-parameter models DeepSeek V3 and R1[11] (DeepSeek-AI et al., 2024; DeepSeek-AI et al., 2025).

Additionally, the supervised fine-tuning (SFT) results on conventional CSC datasets from Li et al. (2024) are also included for better understanding.

Training Details of SFT Baselines The SFT models were trained on a combined dataset consisting of 271k pseudo sentence pairs (Wang et al., 2018) and the **CSCD-NS** training data. This means that while CSCD-NS is an in-domain evaluation, the Lemon dataset serves as a cross-domain dataset for evaluating the generalization capabilities.

5.4 Model Selection

We use the Qwen2.5 model series (Yang et al., 2024) as the main model for our experiments, as it is one of the most recent open-source LLMs with strong Chinese language capabilities.

For ICL experiments, we use the "Instruct" version, which is optimized for instruction follow-

[7] For conventional CSC datasets, exemplars are randomly sampled from the CSCD-NS training set, while for C2EC, they come from its development set. Due to the random nature of ICL, we report the results averaged across 3 runs.

[8] Due to API constraints and cost, instead of beam search, greedy decoding (temperature=0.0) was used, and single-run results are reported.

[9] Version: gpt-4o-mini-2024-07-18

[10] Version: gpt-4o-2024-11-20

[11] Versions: DeepSeek V3 (released on 2024-12-26) and R1 (released on 2025-01-20).

Model	Size	Method	Conventional CSC									C2EC
			Lemon							CSCD-NS	Avg.	C2EC
			Car	*Cot*	*Enc*	*Gam*	*Mec*	*New*	*Nov*	*test*		*test*
Supervised Fine-tuning SoTAs (Li et al., 2024)												
SCOPE	0.1B	Full	50.71	54.89	45.23	24.74	44.44	48.72	33.17	71.70	46.70	–
Qwen1.5	7B	LoRA	53.87	58.04	54.57	37.43	61.16	60.07	41.42	71.64	54.77	–
	14B	LoRA	57.54	60.40	56.48	38.02	65.31	64.49	43.92	**73.80**	57.49	–
Training-free Methods												
GPT4o-mini	N/A	ICL	32.13	29.57	41.43	12.46	34.12	33.53	26.67	40.32	31.28	30.73
GPT4o	N/A	ICL	54.68	57.25	63.02	14.60	62.92	61.37	54.10	65.43	54.17	45.71
DeepSeek V3	671B	ICL	60.12	69.91	**68.56**	38.10	**70.34**	69.67	**61.41**	69.43	63.44	54.62
DeepSeek R1	671B	ICL	57.80	63.47	62.38	45.37	69.38	68.39	58.21	62.68	61.61	48.85
Qwen2.5	7B	ICL	28.98	39.78	36.89	9.89	38.22	27.98	22.09	32.61	29.56	29.03
		ICL-RR	41.39	55.61	49.23	24.13	51.19	44.48	34.14	54.20	44.30	41.01
		TfPf	55.25	64.31	53.89	41.08	57.47	63.39	45.53	62.45	55.42	41.50
		OUR	61.80	**71.05**	65.86	51.67	68.65	69.34	51.89	71.03	63.91	56.02
Qwen2.5	14B	ICL	35.93	49.65	42.04	26.45	45.37	38.02	31.45	38.39	38.41	32.52
		ICL-RR	48.80	55.39	52.63	40.02	53.67	55.46	44.43	55.68	50.76	42.86
		TfPf	55.51	62.50	54.43	37.90	56.58	64.25	46.74	62.53	55.06	40.96
		OUR	**64.62**	70.81	68.50	**51.92**	68.24	**71.85**	53.68	72.75	**65.30**	**57.41**

Table 3: Comparison between our method and the baseline methods on conventional CSC datasets.

ing. As for our approach, we use the "Base" version. To reduce the computational cost, we use 7B variant for detailed analysis.

To further validate the effectiveness of our approach, three other LLM families, Qwen1.5, Baichuan2 and InternLM2, are also discussed in the discussion section.

5.5 Hyperparameters

We largely follow the hyperparameter settings in Zhou et al. (2024), using a beam size of 8 and α for length reward of 2.5. Through development set tuning, we set the weights for insertion and deletion as 8.5 and 9.0, respectively, and the temperature of the prompt-based scoring as 1.5. During the ablation study, we tune the temperature for each setting to maximize its respective F_1 score.

6 Main Results

As shown in Table 3, our method performs better than ICL, ICL-RR, and TfPf on both conventional CSC datasets and the C2EC dataset. Compared to TfPf, from which our method is extended, we achieve improvements of 8.49 and 10.24 on average with 7B and 14B models, respectively. Compared to ICL-RR, our method is shown to be a better way to combine the advantages of prompt-based LLMs and TfPf. This is because ICL-RR can only choose from the top K candidates from ICL. If none of these candidates are good, reranking cannot improve the final result.

Compared to SFT models from Li et al. (2024), which are trained on the training set of the CSCD-NS dataset, our method shows better performance on the **out-of-domain** Lemon dataset. This indicates that SFT methods may not generalize well to new data they have not seen during training. It is also worth noting that, since the SFT models were trained with the Qwen1.5 series, which could make a direct comparison with our method unfair, we also provide our results using the Qwen1.5 series in Appendix E.2.

Compared to recent leading LLMs (e.g., the 671B parameter DeepSeek V3), our method enables much smaller 7B and 14B models to be on par with them without any training.

For a better understanding of the performance of our method, we also provide several qualitative results in Appendix E.4.

An interesting phenomenon worth noting is that the reasoning model DeepSeek R1 shows lower performance than DeepSeek V3, its non-reasoning variant, on both CSC and C2EC tasks. We find this may be caused by incorrect reasoning. More discussion of this is provided in Appendix E.5.

7 Discussion

7.1 Hamming Distance vs. Levenshtein Distance

Recall that in this work, we extend TfPf in two ways. The first is replacing the original Hamming

	Prompt Template	Prompt LLM	Pure LLM	Distance Metric	CSCD-NS *dev*			C2EC *dev*		
					P	**R**	**F_1**	**P**	**R**	**F_1**
1	Minimal	Base	Base	Levenshtein	67.70	73.69	70.57	**65.14**	49.18	56.05
2	Minimal	Base	Base	Hamming	**72.49**	**74.90**	**73.68**	65.06	41.71	50.83
3	–	–	Base	Levenshtein	60.04	65.23	62.53	54.80	37.43	44.48
4	Minimal	Base	–	Levenshtein	61.12	57.91	59.47	55.96	46.18	50.60
5	Detailed	Base	Base	Levenshtein	67.60	71.65	69.57	65.12	45.90	53.85
6	Minimal	Instruct	Instruct	Levenshtein	58.35	69.50	63.44	55.04	50.27	52.55
7	Detailed	Instruct	Instruct	Levenshtein	67.33	68.44	67.88	62.21	**51.73**	56.49

Table 4: Ablation study on different parts of our approach.

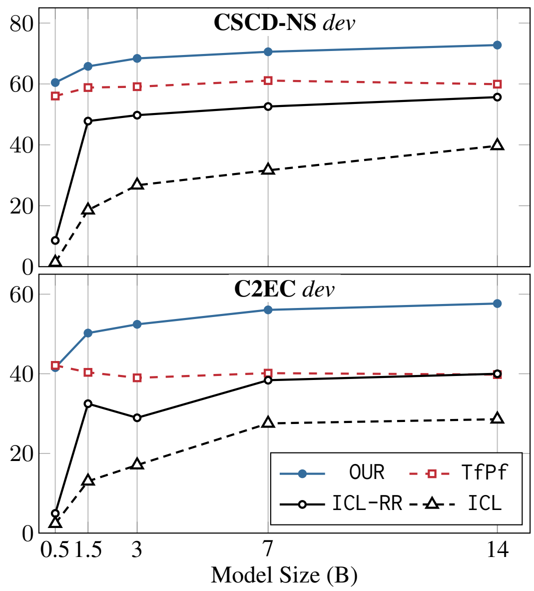


Figure 3: Results of different model sizes.

distance with a Levenshtein distance to allow for insertion and deletion operations.

The second row of Table 4 shows the results when we revert to the original Hamming distance. While the Levenshtein distance-based model performs better on the C2EC dataset, it degrades the performance on the CSCD-NS dataset, the conventional CSC dataset.

This can be attributed to two factors. First, the conventional CSC dataset only contains substitution errors; any insertion or deletion will lead to an over-correction. Second, insertion and deletion operations may compete with replacement operations, leading to mis-corrections. For example, when deletion is allowed, the model may incorrectly delete a misspelled character instead of replacing it with the correct one.

7.2 Effectiveness of Dual-LLM

The second extension is incorporating a prompt-based LLM probability into the `TfPf` framework. The third row of Table 4 shows the results when we remove the prompt-based LLM probability. We observe large performance drops in both precision and recall on both datasets, indicating that the prompt-based LLM probability is a crucial component of our method. An interesting question is *why we still need the pure LLM probability when we already have the prompt-based LLM probability*. The fourth row of Table 4 shows the results when we remove the pure LLM probability. Removing it leads to performance drops of 11.10 and 5.45 points on the CSCD-NS and C2EC datasets, respectively. As pointed out in the reinforcement learning perspective in Appendix C, the pure LLM probability evaluates fluency and correctness from a pure language model perspective, encouraging the model to generate more fluent and correct sentences.

7.3 Impact of Different Prompt Templates

In Section 4.2, we introduce two prompt templates: a minimal prompt and a detailed prompt with more instructions. This section investigates the impact of different prompt templates. Rows 5-7 of Table 4 show the results of using the minimal prompt and the detailed prompt with the base LLM and instruction-tuned LLM. The results show that base LLM favors the minimal prompt, while instruction-tuned versions prefer the longer and more complex prompt. Compared to instruction-tuned versions, base LLM is more robust to prompt variations.

7.4 Impact of the LLM Size

To investigate the impact of model size, we conduct experiments across different sizes of the Qwen2.5 series. The results are shown in Figure 3. We observe that our method's performance generally improves with increasing model size. In contrast, `TfPf`'s performance does not consistently improve, indicating that our method better leverages the scale of larger LLMs. However, on the 0.5B model,

Model	Method	CSCD-NS *dev*			C2EC *dev*		
		P	**R**	**F_1**	**P**	**R**	**F_1**
`Qwen2.5 7B`	`ICL`	24.11	46.07	31.65	20.55	41.74	27.54
	`ICL-RR`	47.83	58.39	52.59	33.24	45.45	38.39
	`TfPf`	57.98	64.61	61.11	52.03	32.70	40.16
	`OUR`	**67.70**	**73.69**	**70.57**	**65.14**	**49.18**	**56.05**
`Qwen1.5 7B`	`ICL`	9.32	43.12	15.33	7.40	40.56	12.51
	`ICL-RR`	18.50	53.68	27.51	10.59	44.87	17.13
	`TfPf`	52.42	60.69	56.25	51.05	31.06	38.62
	`OUR`	**62.77**	**71.03**	**66.64**	**62.65**	**43.08**	**51.05**
`Baichuan2 7B`	`ICL`	12.41	36.36	18.49	7.79	26.02	11.99
	`ICL-RR`	27.65	48.42	35.19	15.05	29.90	20.01
	`TfPf`	67.96	**62.96**	65.37	58.21	29.69	39.32
	`OUR`	**70.87**	61.16	**65.66**	**62.23**	**34.06**	**44.03**
`InternLM2 7B`	`ICL`	22.16	39.47	28.38	15.56	30.90	20.43
	`ICL-RR`	24.97	39.83	30.70	23.74	31.75	27.17
	`TfPf`	61.12	63.08	62.08	49.92	26.78	34.86
	`OUR`	**65.71**	**68.29**	**66.97**	**60.59**	**39.62**	**47.91**

Table 5: Results of applying our method to different LLM families. All models are with 7B parameters.

which has very limited prompt understanding, `TfPf` may outperform our method.

7.5 Impact of Different LLM Families

To investigate if our method can improve the performance of other LLMs, we conducted experiments on three additional LLMs: `Qwen1.5` (Bai et al., 2023), `Baichuan2` (Yang et al., 2023a), and `InternLM2` (Cai et al., 2024). For these LLMs, we used the 7B variant.

The results are shown in Table 5. Our method consistently improves the performance of these LLMs. However, the performance gain with `TfPf` varies across different LLMs. We observed that the performance improvement might be related to the recall value of `ICL` for different LLMs. A higher recall value of `ICL` corresponds to a larger performance improvement with our method. We plan to explore the underlying mechanism of this phenomenon in the future.

7.6 Performance on Different Error Types

Figure 4 shows the performance across different error types. Our method outperforms baselines on all error types. The difficulty levels of different error types vary. Substitution (misspelling) errors are the easiest to correct, with all methods achieving relatively high performance. Redundant errors are the most challenging. This is mainly because models tend to add optional characters to the error-free sentence, leading to over-correction. An example of this is given in Appendix E.4.

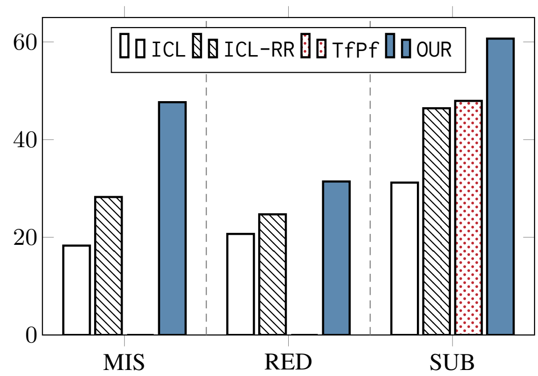


Figure 4: The results of different types of error on the C2EC dev set. MIS: Missing, RED: Redundant, SUB: Substitution (Misspelling)

7.7 More Discussion

Due to space limitations, some interesting discussions are included in Appendix F. These include: *a)* Investigation of beam size impact (F.1), *b)* Ablation study on length reward and faithfulness reward adopted from `TfPf` (F.2), *c)* A fair comparison with supervised fine-tuning on `Qwen1.5` series (E.2) *d)* How to adjust the precision-recall trade-off (F.3), and *e)* Runtime analysis (F.4).

8 Related Works

8.1 Datasets of Chinese Spelling Correction

Chinese Spelling Correction (CSC) has long been an important research area in NLP, with new datasets continuously being developed to address various needs. The Sighan series (Wu et al., 2013;

Yu et al., 2014; Tseng et al., 2015) is one of the earliest and most influential collections of CSC datasets. While widely used, they are criticized for their poor annotation quality, limited domain, and unrealistic error patterns (Yang et al., 2023b; Wu et al., 2023; Sun et al., 2024).

While researchers like (Yang et al., 2023b; Sun et al., 2024) have tried to improve Sighan datasets through re-annotation, the inherent issues of unrealistic domain and limited error patterns remain. To address these issues, researchers manually created errors across financial, official documents and medical domains (Jiang et al., 2022; Lv et al., 2023). Instead of creating errors from correct sentences, Wu et al. (2023) collected and annotated real errors from seven different domains, building a large dataset that challenges models by not providing training data. Similarly, Hu et al. (2024) created a new dataset with real errors found on social media.

Some works have tried to broaden the scope of conventional CSC datasets. Similar to our work, He et al. (2023) also pointed out that existing datasets mainly focus on substitution errors, overlooking two other common types: insertion and deletion errors, limiting the practicality of the task. To address this, they created the ECMR-2023 dataset by randomly adding, removing, or replacing characters in correct sentences. However, these artificially generated errors may not reflect real-world mistakes well. In contrast, we build our C2EC dataset using existing Chinese text correction data, providing a more realistic benchmark for general Chinese character error correction.

8.2 Approaches of Chinese Spelling Correction

For many years, BERT-based models have dominated CSC research (Zhang et al., 2020; Xu et al., 2021; Zhu et al., 2022; Liang et al., 2023). While these models show strong performance on in-domain datasets, recent studies reveal their limitations when applied to different domains (Wu et al., 2023; Liu et al., 2024c).

With the advent of LLMs, researchers have begun exploring their potential for CSC.

Early attempts focused on prompt-based methods (Li et al., 2023). For instance, Dong et al. (2024) enhanced prompts with pronunciation and glyph information. However, lightweight LLMs, such as those with 7B or 14B parameters, still struggle to achieve satisfactory performance with prompt-based methods. Compared to prompt-based methods, supervised fine-tuning methods have shown more effectiveness. A representative work is Li et al. (2024), which introduced a novel training paradigm that retrains LLMs at the character level to better align with CSC requirements.

Recently, Zhou et al. (2024) proposed `TfPf`, a training-free and prompt-free framework that achieves comparable cross-domain performance to `SFT` methods. As detailed in §3, `TfPf` combines a language model with a distance metric to balance fluency and minimal edits. Our work builds upon this elegant framework by improving its performance and extending it to handle the C2EC task.

9 Conclusion

In this work, we propose the task of C2EC, which is an extended CSC task that handles misspellings and two previously ignored errors: redundant and missing characters. To support this task, we construct a focused C2EC dataset containing real-world errors by combining and manually verifying two existing datasets. We propose a training-free method for the C2EC task by extending the `TfPf` method. Experiments show that our method achieves large improvements over both 10-shot in-context learning and the `TfPf` baseline on conventional CSC and C2EC datasets, achieving competitive performance with models nearly 50 times larger.

Limitations

There are several limitations in this work that we plan to investigate and address in future work.

Research Scope The scope of this study is limited to Chinese character error correction. However, we believe that with some modifications, our approach can be applied to other languages and more complex tasks like grammatical error correction and sentence simplification. We plan to extend our approach to these areas in future work.

In this paper, we focus on evaluating LLM performance during inference under zero-shot or few-shot scenarios. When sufficient annotated data is available, how to effectively utilize it to improve performance of our approach is an interesting question we plan to explore in future work.

Moreover, limited by computational resources, we have not tested our approach on larger models. We believe that models with more parameters would also benefit from our approach.

Speed and Resources Our approach relies on large language models, which inherently demand significant computational resources. Additionally, the inference speed is constrained by the requirement of two forward passes of the LLM at each generation step. Optimizing the implementation, for instance, by ensuring compatibility with modern LLM inference frameworks, might leverage the latest advancements in LLM inference to improve the speed of our approach.

Flexibility Our approach requires access to the model's probability distribution, which makes it unable to be directly applied to API-accessed models that are often more powerful. However, given the flexibility of prompt-based LLMs, we believe there is potential to leverage results from API-accessed models to further improve performance.

Ethics Statement

Our proposed dataset is built upon existing publicly available datasets. We have properly cited the original datasets in our paper and ensured that our use is consistent with their original intent. For works that use our dataset, we require them to appropriately cite the original datasets.

For this work, we manually verified the dataset to ensure quality. We recruited four graduate students who are native Chinese speakers with high Chinese proficiency. The verification process took about 16 working hours per annotator. Each annotator was compensated at a rate of ¥25 per hour.

Acknowledgments

First of all, we would like to express our sincere gratitude to the anonymous reviewers for their valuable suggestions and comments.

We sincerely thank Haozhe Zhou, Ziheng Qiao, Haochen Jiang, and Yumeng Liu for their manual verification of the dataset. We are also deeply grateful to Chen Gong for her continuous support throughout this research.

This work was supported by National Natural Science Foundation of China (Grant No. 62261160648 and 62176173) and a project funded by the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institutions.

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.

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.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv preprint*, abs/2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, and Wangding Zeng. 2024. Deepseek-v3 technical report. *ArXiv preprint*, abs/2412.19437.

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.

Zheyu He, Yujin Zhu, Linlin Wang, and Liang Xu. 2023. UMRSpell: Unifying the detection and correction parts of pre-trained models towards Chinese missing, redundant, and spelling correction. In *Proceedings of ACL*, pages 10238–10250, Toronto, Canada.

Yong Hu, Fandong Meng, and Jie Zhou. 2024. CSCD-NS: a Chinese spelling check dataset for native speakers. In *Proceedings of ACL*, pages 146–159, Bangkok, Thailand.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll L. Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, and Dane Sherburn. 2024. Gpt-4o system card. *ArXiv preprint*, abs/2410.21276.

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.

Kunting Li, Yong Hu, Liang He, Fandong Meng, and Jie Zhou. 2024. C-LLM: learn to check chinese spelling errors character by character. *ArXiv preprint*, abs/2406.16536.

Yinghui Li, Haojing Huang, Shirong Ma, Yong Jiang, Yangning Li, Feng Zhou, Hai-Tao Zheng, and Qingyu Zhou. 2023. 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.

Changchun Liu, Kai Zhang, Junzhe Jiang, Zirui Liu, Hanqing Tao, Min Gao, and Enhong Chen. 2024a. ARM: An alignment-and-replacement module for Chinese spelling check based on LLMs. In *Proceedings of EMNLP*, pages 10156–10168, Miami, Florida, USA.

Linfeng Liu, Hongqiu Wu, and Hai Zhao. 2024b. Chinese spelling correction as rephrasing language model. In *Proceedings of the AAAI*, pages 18662–18670.

Linfeng Liu, Hongqiu Wu, and Hai Zhao. 2024c. Chinese spelling correction as rephrasing language model. *Proceedings of AAAI*, 38(17):18662–18670.

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).

Ziheng Qiao, Houquan Zhou, Yumeng Liu, Zhenghua Li, Min Zhang, Bo Zhang, Chen Li, Ji Zhang, and Fei Huang. 2024. DISC: plug-and-play decoding intervention with similarity of characters for chinese spelling check. *ArXiv preprint*, abs/2412.12863.

Changxuan Sun, Linlin She, and Xuesong Lu. 2024. Two issues with Chinese spelling correction and a refinement solution. In *Proceedings of ACL*, pages 196–204, Bangkok, Thailand.

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.

Baoxin Wang, Xingyi Duan, Dayong Wu, Wanxiang Che, Zhigang Chen, and Guoping Hu. 2022. CCTC: A cross-sentence Chinese text correction dataset for native speakers. In *Proceedings of COLING*, pages 3331–3341, Gyeongju, Republic of Korea.

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.

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. *ArXiv preprint*, abs/2309.10305.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *ArXiv preprint*, abs/2412.15115.

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.

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.

Honghong Zhao, Baoxin Wang, Dayong Wu, Wanxiang Che, Zhigang Chen, and Shijin Wang. 2022. Overview of CTC 2021: Chinese text correction for native speakers. *ArXiv preprint*, abs/2208.05681.

Houquan Zhou, Zhenghua Li, Bo Zhang, Chen Li, Shaopeng Lai, Ji Zhang, Fei Huang, and Min Zhang. 2024. A simple yet effective training-free prompt-free approach to Chinese spelling correction based on large language models. In *Proceedings of EMNLP*, pages 17446–17467, Miami, Florida, USA.

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.

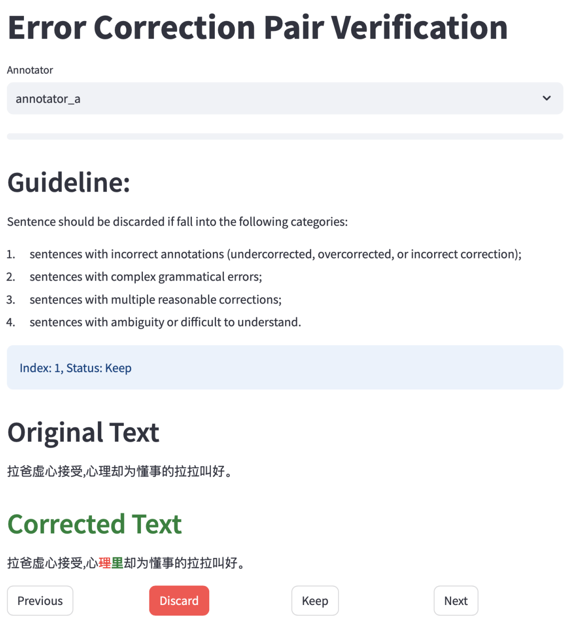


Figure 5: The annotation UI for data verification.

	Incorrect annotations
Input	特种兵那么厉害,一旦犯罪了怎么组织?其实国家早有防备
Annotation	特种兵那么厉害,一旦犯罪了怎么组止?其实国家早有防备
	Having complex grammatical errors
Input	千佛山起源于济南南部山区，山势宛如巨大的一轮秋月，弯伏恢宏。
Annotation	千佛山起源于济南南部山区，山势宛如一轮巨大的秋月，弯伏恢宏。
	Having multiple reasonable corrections
Input	二人此时此刻,站在一起,令人四周不少人,心中羡慕。
Annotation	二人此时此刻,站在一起,令得四周不少人,心中羡慕。
	Ambiguous or difficult to understand sentences
Input	薛静妃点头,“我想要带着整个阴阳宗去混沌妖族!”
Annotation	薛静妃点头,“我想要带着整个阴阳宗去混沌宇宙!”

Table 6: Some examples of the sentences that are discarded during the manual verification.

A Details of C2EC Dataset

A.1 Annotation UI for Data Verification

Figure 5 shows our verification user interface. To maintain verification quality, we always present the guidelines to the annotators at the top of the UI. Additionally, we highlight the differences between the original and corrected sentences to help the annotators easily identify the changes. We also require the annotators to carefully remove any sentences containing sensitive or offensive content. However, we did not find any such sentences during verification.

A.2 Discarded Cases

We present examples of sentences discarded during manual verification in Table 6. The first example involves a sentence with incorrect annotation. The input sentence incorrectly spells the word “阻止” (prevent, *zǔ zhǐ*) as “组织” (organize, *zǔ zhī*), but the annotator from the original dataset only corrected the character to “止”, leaving “组” unchanged. The second example is a sentence with complex grammatical errors. In Chinese textbooks, there is a recommended rule for arranging multiple adjectives in a sentence for a formal style. For instance, adjectives indicating time or location should precede those of quantity, and quantity adjectives should precede attributive adjectives. Thus, a grammatically correct phrase in this context is “一轮巨大的秋月”. However, correcting such sentences is beyond the scope of this work. The third example is a sentence with multiple reasonable corrections. The incorrect character “人” (person, *rén*) can be corrected by either removing it or changing it to “得” (an auxiliary verb, *dé*). The fourth example is an ambiguous sentence that requires additional context for accurate correction. Given the context, we cannot determine whether the term “混沌妖族” (*Chaos Demon Clan*, *hùn dùn yāo zú*) is valid within the novel, or if it should be corrected to another term, leading us to discard this sentence.

We plan to re-annotate discarded sentences in the future and share them with the original authors to improve the dataset.

B Distance Metrics

B.1 Hamming Distance

Hamming distance measures the position-wise differences between two strings of equal length:

$$\texttt{Dist}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} w(x_i, y_i) \tag{5}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are strings of the same length n, and $w(x_i, y_i)$ is the weight assigned to the characters x_i and y_i. In the standard scenario, $w(x_i, y_i) = 1$ if $x_i \neq y_i$ and 0 otherwise. For instance, the Hamming distance between `abcdef` and `a1c23f` is 3.

Distortion Model of TfPf as a Weighted Hamming Distance The distortion model $\log p_{\mathtt{DM}}(\boldsymbol{x} \mid \boldsymbol{y})$ in TfPf is character-level factorized:

$$\log p_{\mathtt{DM}}(\boldsymbol{x} \mid \boldsymbol{y}) = \sum_{i=1}^{n} \log p_{\mathtt{DM}}(x_i \mid y_i) \tag{6}$$

By interpreting $-\log p_{\mathtt{DM}}(x_i \mid y_i)$ as the weight for characters x_i and y_i, the distortion model of TfPf can be seen as a weighted Hamming distance.

B.2 Levenshtein Distance

Levenshtein distance measures the difference between two strings, which may have different lengths, by calculating the minimum number of edit operations (substitution, insertion, and deletion) required. For example, the Levenshtein distance between abcde and a1bc2e is 2, including an insertion of '1' between 'a' and 'b', and a substitution of 'd' with '2'.

The Levenshtein distance can be computed using a dynamic programming algorithm with the following recurrence relation:

$$\begin{aligned} &\mathtt{Dist}(\boldsymbol{x}_{\leqslant m}, \boldsymbol{y}_{\leqslant n}) \\ &= \min \begin{cases} \mathtt{Dist}(\boldsymbol{x}_{\leqslant m-1}, \boldsymbol{y}_{\leqslant n}) + w_{\mathtt{I}} \\ \mathtt{Dist}(\boldsymbol{x}_{\leqslant m}, \boldsymbol{y}_{\leqslant n-1}) + w_{\mathtt{D}} \\ \mathtt{Dist}(\boldsymbol{x}_{\leqslant m-1}, \boldsymbol{y}_{\leqslant n-1}) + w_{\mathtt{S}}(x_m, y_n) \end{cases} \end{aligned}$$

where $w_{\mathtt{I}}$, $w_{\mathtt{D}}$, and $w_{\mathtt{S}}(x_m, y_n)$ are the weights for insertion, deletion, and substitution, respectively. For simplicity, the keep operation is treated as a special case of substitution.

C A Reinforcement Learning Perspective

In this section, we provide a reinforcement learning perspective of our method. We believe this perspective can help us understand the role of the two large language models in our method, and the relationship between our method and TfPf.

C.1 KL-Regulated Reinforcement Learning

Reinforcement learning (RL) has been widely used to improve the performance of LLMs by optimizing the following objective:

$$\begin{aligned} \mathcal{L}(p_\theta) = \mathbb{E}_{\boldsymbol{y} \sim p_\theta(\cdot \mid \boldsymbol{x})} \left[r(\boldsymbol{x}, \boldsymbol{y}) \right] \\ - \beta D_{KL} \left(p_\theta (\cdot \mid \boldsymbol{x}) \parallel p_{\mathtt{ref}} (\cdot \mid \boldsymbol{x}) \right) \end{aligned} \tag{7}$$

where p_θ is the model we want to optimize, r is the reward function, $p_{\mathtt{ref}}$ is the reference model whose parameters are frozen, and β is the KL-regularization coefficient that controls how much we want p_θ to be different from the reference model. A larger β means we want p_θ to be more similar to the reference model.

The optimal model p^* of p_θ is unique and is given by:

$$p^*(\boldsymbol{y} \mid \boldsymbol{x}) \propto p_{\mathtt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) \exp\left(\frac{1}{\beta} r(\boldsymbol{x}, \boldsymbol{y})\right) \tag{8}$$

This equation shows that we do not need to train the model to optimize the objective; instead, we can directly obtain the optimal policy by combining the reference model and the reward function during inference.

C.2 TfPf as a Reward Model

Intuitively, given an input $\boldsymbol{x}$ and two outputs $\boldsymbol{y}_a$ and $\boldsymbol{y}_b$, which output is better can be determined by the following two criteria:

- **Fluency**: A better output $\boldsymbol{y}$ should be more fluent.
- **Faithfulness**: A better output $\boldsymbol{y}$ should only make the necessary changes to the input sentence $\boldsymbol{x}$. Between two sentences with the same fluency, the one with fewer changes is better.

Recall the score function of TfPf in Equation 1. The score function of TfPf is a combination of a large language model and a distance metric. The large language model, acting as a pure language model, can be seen as a reward function measuring the fluency of the output. On the other hand, the distance metric, acting as a reward function measuring the faithfulness of the output, penalizes outputs that change the input sentence too much.

Combining TfPf as a reward model with the RL framework, Equation 8 can be rewritten as:

$$\begin{aligned} \log p^*(\boldsymbol{y} \mid \boldsymbol{x}) \propto \ & \log p_{\mathtt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) \\ & + \frac{1}{\beta} \left(\begin{array}{c} \log p_{\mathtt{LLM}}(\boldsymbol{y}) \\ - \\ \mathtt{Dist}(\boldsymbol{x}, \boldsymbol{y}) \end{array} \right) \end{aligned} \tag{9}$$

Link to the Original TfPf Paper If we set $\beta = 1$ and use a uniform distribution as the reference model $\log p_{\mathtt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) = C$, where C is a constant, the above equation is equivalent to the score function of TfPf in Equation 1.

Link to Our Method If we set $\beta = 1$ and use a prompt-based large language model as the reference model $\log p_{\mathtt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) = \log p_{\mathtt{LLM}}(\boldsymbol{y} \mid$

$\texttt{Prompt}(\boldsymbol{x})$), the above equation becomes equivalent to our score function in Equation 2. By using a prompt-based large language model as the reference model instead of a uniform distribution, we obtain a more reasonable prior distribution, which enables our method to find better outputs $\boldsymbol{y}$ with a limited beam size.

C.3 Proof of Equation 8

First, we rewrite the objective function of RL as follows:

$$\begin{aligned}\mathcal{L}(p_\theta) &= \mathbb{E}_{\boldsymbol{y}\sim p_\theta(\cdot|\boldsymbol{x})}\left[r(\boldsymbol{x},\boldsymbol{y})\right] \\ &\quad - \beta D_{KL}\left(p_\theta\left(\cdot \mid \boldsymbol{x}\right) \| \, p_{\texttt{ref}}\left(\cdot \mid \boldsymbol{x}\right)\right) \\ &= \sum_{\boldsymbol{y}} p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) \begin{pmatrix} r(\boldsymbol{x},\boldsymbol{y}) \\ - \\ \beta \log \frac{p_\theta(\boldsymbol{y}|\boldsymbol{x})}{p_{\texttt{ref}}(\boldsymbol{y}|\boldsymbol{x})}\end{pmatrix}\end{aligned} \tag{10}$$

Then we can compute the gradient of the objective function with respect to the policy $p_\theta(\boldsymbol{y} \mid \boldsymbol{x})$:

$$\frac{\partial \mathcal{L}(p_\theta)}{\partial p_\theta(\boldsymbol{y} \mid \boldsymbol{x})} = r(\boldsymbol{x},\boldsymbol{y}) - \beta\left(\log \frac{p_\theta(\boldsymbol{y} \mid \boldsymbol{x})}{p_{\texttt{ref}}(\boldsymbol{y} \mid \boldsymbol{x})} + 1\right) \tag{11}$$

Now, we can find the optimal policy p^* by setting the gradient of the objective function to zero:

$$r(\boldsymbol{x},\boldsymbol{y}) - \beta\left(\log \frac{p_\theta(\boldsymbol{y} \mid \boldsymbol{x})}{p_{\texttt{ref}}(\boldsymbol{y} \mid \boldsymbol{x})} + 1\right) = 0 \tag{12}$$

By rearranging the equation and taking the exponential of both sides, we get:

$$\begin{aligned}p^*(\boldsymbol{y} \mid \boldsymbol{x}) &= p_{\texttt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) \exp\left(\frac{r(\boldsymbol{x},\boldsymbol{y})}{\beta} - 1\right) \\ &\propto p_{\texttt{ref}}(\boldsymbol{y} \mid \boldsymbol{x}) \exp\left(\frac{r(\boldsymbol{x},\boldsymbol{y})}{\beta}\right)\end{aligned} \tag{13}$$

where the -1 term inside the exponential function can be safely ignored because it is a constant.

D Detailed Experiment Settings

D.1 Prompt Templates

The Detailed Prompt for Our Method Figure 6 shows the detailed prompt for our method, which includes task-specific system prompts, format requirements, and additional notes.

Prompts for In-Context Learning Baselines The prompts for the ICL baseline are shown in Figure 13. To ensure the performance of the ICL baseline, we use different prompts for the ICL baseline on conventional CSC and C2EC. Specifically, when evaluating conventional CSC, we do not instruct the model to correct missing and redundant characters.

D.2 An Approximate Implementation of Our Method

To reduce computational cost and speed up the generation process, we make two approximations. First, we approximate the incremental Levenshtein distance $\Delta_{\texttt{Dist}_\texttt{L}}(\boldsymbol{x}, a, b, \boldsymbol{t}_1 \ldots \boldsymbol{t}_{k-1}, \boldsymbol{t}_k)$ as $\texttt{Dist}_\texttt{L}(\boldsymbol{x}_{[a:b+1]}, \boldsymbol{t}_k)$. Then, we reduce the search space by maintaining only one best end index b for each $\boldsymbol{t}_1 \ldots \boldsymbol{t}_k$. While these approximations may yield suboptimal results, they work well in practice.

D.3 Dataset Statistics

In this work, we use three datasets to evaluate the performance of our method. All datasets used in this work are publicly available. Specifically, the CSCD-NS dataset is publicly available under the MIT license, while the CCTC dataset is publicly available under the Apache 2.0 license. The specifics of these datasets are listed in Table 7. The **Evaluation Sentences** row in Table 7 shows the number of sentences actually used for evaluation. This is because sentences where the original and corrected versions differ in length are excluded when evaluating CSC models, as done in previous works (Liu et al., 2024b; Li et al., 2024; Zhou et al., 2024).

D.4 Evaluation Implementation and Settings

We use the evaluation script from Zhou et al. (2024) to calculate the metrics. This script adopts a Levenshtein distance algorithm to extract edit operations, allowing for comparisons between sentences of different lengths. We also follow their settings by ignoring whitespaces and converting all full-width punctuation to half-width.

D.5 Hardware Setup

Experiments were conducted on a single NVIDIA A100 40GB GPU with the Intel Xeon Gold 6248R (3.00GHz) CPU.

E More Results

E.1 Sentence-level Results

The sentence-level F_1 results are shown in Table 8.

The results show that our method outperforms three baselines on both conventional CSC and C2EC datasets.

Datasets	Lemon							CSCD-NS		C2EC	
Subsets	*Car*	*Cot*	*Enc*	*Gam*	*Mec*	*New*	*Nov*	*dev*	*Test*	*dev*	*Test*
Language	Chinese										
All Sentences	3,410	1,026	3,434	400	2,090	5,892	6,000	5,000	5,000	1,995	5,711
Evaluation Sentences	3,245	993	3,274	393	1,942	5,887	6,000	5,000	5,000	1,995	5,711
Erroneous Sentence Ratio	48.60	44.41	48.59	37.66	46.60	49.96	50.23	46.28	46.06	50.08	49.92
Average Length	43.44	40.11	39.95	32.81	39.18	25.15	36.24	57.45	57.63	51.93	41.88
Average Error Character	1.20	1.10	1.12	1.10	1.13	1.11	1.13	1.10	1.10	1.10	1.16

Table 7: The statistics of the datasets used in the experiments.

Model	Size	Method	Conventional CSC									C2EC
			Lemon							CSCD-NS	Avg.	C2EC
			Car	*Cot*	*Enc*	*Gam*	*Mec*	*New*	*Nov*	*test*		*test*
Supervised Fine-tuning SoTAs (Wu et al., 2023; Liu et al., 2024c; Hu et al., 2024)												
BERT	0.1B	SFT	52.3†	64.1†	45.5†	33.3†	50.9†	56.0†	36.0†	72.96‡	–	–
SM-BERT	0.1B	SFT	52.0†	65.0†	44.6†	29.8†	49.3†	55.8†	37.8†	**73.62**‡	–	–
ReLM	0.1B	SFT	53.1†	66.8†	49.2†	33.0†	54.0†	58.5†	37.8†	–	–	–
Baichuan2	7B	SFT	–	–	–	–	–	–	–	64.44‡	–	–
	13B	SFT	–	–	–	–	–	–	–	66.10‡	–	–
BERT-based SFT Models cooperating with LLMs (Liu et al., 2024a)												
MDCSPell$^{\natural}$	N/A	ARM	37.1	52.7	35.2	15.3	33.0	36.4	15.6	–		
Training-free Methods of LLMs												
GPT4o-mini	N/A	ICL	29.43	30.85	42.11	26.67	33.72	33.12	23.52	38.78	32.27	29.09
GPT4o	N/A	ICL	52.91	59.98	62.15	39.11	61.87	61.82	49.96	63.89	56.46	47.66
DeepSeek V3	671B	ICL	57.33	**69.87**	**66.07**	**53.12**	**69.33**	69.14	**57.43**	67.55	**63.73**	53.25
DeepSeek R1	671B	ICL	54.52	63.43	59.93	49.27	68.59	67.82	52.93	62.39	59.84	45.58
Qwen2.5	7B	ICL	26.41	42.95	37.97	18.93	38.27	29.10	21.85	34.10	31.20	28.53
		ICL-RR	38.93	56.46	49.05	32.40	50.31	44.54	32.77	54.40	44.86	40.24
		TfPf	49.37	61.60	48.48	39.07	55.69	59.79	39.09	58.57	51.46	38.60
		OUR	57.58	69.45	61.56	47.78	66.59	66.72	47.91	68.13	60.72	52.68
Qwen2.5	14B	ICL	35.91	52.31	43.55	32.86	47.95	40.70	31.59	41.54	40.80	31.73
		ICL-RR	48.15	59.69	53.99	37.19	55.60	57.02	42.38	59.30	51.67	42.42
		TfPf	50.25	59.92	49.84	32.88	54.39	60.94	41.11	57.95	50.91	37.84
		OUR	**59.83**	68.74	64.37	48.53	66.54	**69.64**	49.65	70.42	62.22	**54.12**

Table 8: Sentence-level F_1 scores of our method and the baseline methods on conventional CSC datasets. † indicates the results are from models fine-tuned on 34M synthetic CSC data (Wu et al., 2023; Liu et al., 2024c). ‡ indicates the results are from models pre-trained on 2M synthetic data specifically designed for the CSCD-NS dataset before fine-tuning on the CSCD-NS training set (Hu et al., 2024). ♮ The ARM method is based on the GPT3.5 Turbo model and a BERT-based model (MDCSPell), which is trained on 271k synthetic CSC data from Wang et al. (2018) and training data from the Sighan series datasets.

Detailed Prompt

System Prompt:
你是一个优秀的中文纠错模型，中文纠错模型即更正用户输入句子中的错误。\n
User Prompt:
你需要识别并纠正用户输入的句子中可能的错别字、多字、漏字并输出正确的句子，在修改的同时尽可能减少对原句子的改动(不新增、删除和修改标点符号)。只输出没有错误的句子，不要添加任何其他解释或说明。如果句子没有错误，就直接输出和输入相同的句子。\n
输入：{INPUT: x}\n

System Prompt:
You are an excellent Chinese error correction model, which means you correct errors in the sentences entered by the user.\n
User Prompt:
You need to identify and correct possible misspellings, redundant and missing characters in the user's input sentence, and output the correct sentence. While making corrections, try to minimize changes to the original sentence (without adding, deleting, or modifying punctuation). Only output the corrected sentence; do not add any further explanations or notes. If the sentence is error-free, output the exact same sentence as the input.\n
Input: {INPUT: x}\n

Figure 6: A detailed prompt template.

Additionally, our method enables a 14B model to achieve performance comparable to the leading LLM, which has 671B parameters.

E.2 A Fair Comparison with SFT Methods

In main results, we compare our approach using the Qwen2.5 series with the state-of-the-art methods that have been fine-tuned with supervision, as reported by (Li et al., 2024). It's important to note a potential discrepancy: their supervised fine-tuned methods were trained from the Qwen1.5 series, whereas our method utilizes the Qwen2.5 series. To ensure a fair comparison, we also provide the results of our method on the Qwen1.5 series in Table 9.

When compared to the TfPf method, the SFT methods show superior performance on the in-domain dataset CSCD-NS. However, they perform less effectively on the out-of-domain dataset Lemon, particularly with a 7B model. This suggests that the SFT methods might overfit to the in-domain dataset CSCD-NS, limiting their generalization to the out-of-domain dataset Lemon.

Our method, which requires no training, significantly outperforms the SFT methods on the out-of-domain dataset Lemon. Additionally, without any training, our method achieves performance on par with the SFT methods on the in-domain dataset CSCD-NS, with scores of 71.53 versus 73.80 on the 14B model.

E.3 Additional Comparison on Sighan dataset

To provide a more comprehensive comparison, we also report the results of our method on the Sighan dataset, a CSC dataset, as well as some classical CSC models.

The results are shown in Table 10. From the results, we can see that our method still lags behind classical CSC models, which are specifically trained on the Sighan dataset.

E.4 Qualitative Analysis

Table 11 shows four examples illustrating the effectiveness of our method.

In the first example, characters "曲安" (*qū ān*) were incorrectly typed as "圈" (*quān*). This simple error highlights the limitation of the Hamming distance in TfPf, which led to an incorrect correction to "甲" (*jiǎ*) instead of the correct insertion of a character.

The second example is the character "调" (*diào*) in "调方" (*prescription adjustment*), which was mistakenly entered as "凋" (*diāo*). Correcting this error requires subsequent contextual understanding, which pure LLM probabilities in TfPf failed to achieve, resulting in an incorrect high-frequency substitution to "调换" (*exchange, diào huàn*). This issue was mitigated by incorporating prompt-based LLM probabilities.

In the third example, the character "乙" (*A, yǐ*) was mistyped as "以" (*yǐ*). The ICL baseline overlooked phonetic similarities, incorrectly changing it to "甲" (*A, jiǎ*). The ICL-RR baseline also failed to correct this error as the correction "以"→"乙" was not among the top-K candidates.

The fourth example is a negative case where our method unnecessarily inserted an "以" into an already correct sentence. While this change

Model	Size	Method	Lemon *Car*	*Cot*	*Enc*	*Gam*	*Mec*	*New*	*Nov*	CSCD-NS *test*	Avg.
SCOPE	0.1B	SFT-F	50.71	54.89	45.23	24.74	44.44	48.72	33.17	71.70	46.70
Qwen1.5	7B	SFT-L	53.38	56.55	54.44	37.33	59.21	58.96	39.12	68.66	53.46
		SFT-L†	53.87	58.04	54.57	37.43	61.16	60.07	41.42	**71.64**	54.77
		TfPf	53.88	61.68	51.46	38.87	57.66	60.97	44.97	58.27	53.47
		OUR	**61.57**	**69.10**	**63.34**	**48.50**	**65.34**	**68.89**	**50.27**	67.25	**61.78**
Qwen1.5	14B	SFT-L	54.56	56.82	53.44	32.59	58.89	63.32	40.58	72.63	54.10
		SFT-L†	57.54	60.40	56.48	38.02	65.31	64.49	43.92	**73.80**	57.49
		TfPf	52.61	62.91	50.81	36.36	54.78	60.59	42.89	58.56	52.44
		OUR	**62.88**	**70.31**	**66.24**	**46.59**	**66.67**	**70.15**	**52.69**	71.53	**63.38**

Table 9: Fair comparison between our method and the supervised fine-tuning (SFT) methods. We adopt the SFT method from Li et al. (2024). SFT-F means the full parameter fine-tuning, while SFT-L means fine-tuning with LoRA. SFT-L† means the C-LLM method from Li et al. (2024), that conducts the Character-level LoRA fine-tuning after the continuous pre-training.

Models	Sighan *13* Detection P	R	F_1	Correction P	R	F_1	Sighan *14* Detection P	R	F_1	Correction P	R	F_1	Sighan *15* Detection P	R	F_1	Correction P	R	F_1
Supervised Fine-tuning SoTAs (reported by Qiao et al. (2024))																		
SpellGCN	80.1	74.4	77.2	78.3	72.7	75.4	65.1	69.5	67.2	63.1	67.2	65.3	74.8	80.7	77.7	72.1	77.7	75.9
ReaLiSe	88.6	82.5	85.4	87.2	81.2	84.1	67.8	71.5	69.6	66.3	70.0	68.1	77.3	81.3	79.3	75.9	79.9	77.8
+DISC	88.9	82.2	85.4	87.6	81.1	84.2	69.2	71.2	70.1	68.2	70.2	69.2	78.3	81.2	79.7	77.0	79.9	78.4
SCOPE	87.5	83.0	85.2	86.5	82.1	84.2	68.8	73.7	71.1	67.1	71.2	69.5	80.5	85.4	82.9	78.7	83.5	81.0
+DR-CSC	88.5	83.7	86.0	87.7	83.0	85.3	**70.2**	73.3	71.7	69.3	72.3	70.7	**82.9**	84.8	**83.8**	**80.3**	82.3	81.3
+DISC	88.8	83.7	86.2	88.0	83.0	85.4	**70.2**	73.5	71.8	**69.3**	72.5	70.9	81.7	84.8	83.2	80.2	83.4	**81.8**
ReLM	86.4	83.7	85.0	85.0	82.3	83.7	65.7	74.5	69.8	63.7	72.3	67.7	78.3	**85.6**	81.8	76.8	**83.9**	80.2
+DISC	**89.7**	**84.5**	**87.0**	**88.4**	**83.3**	**85.8**	69.7	**74.9**	**72.2**	68.6	**73.7**	**71.0**	80.8	84.3	82.5	79.8	83.1	81.4
Training-free Methods																		
GPT3.5	61.6	29.2	39.7	57.1	27.1	36.7	41.4	23.1	29.6	39.7	22.1	28.4	39.4	46.4	42.6	32.7	38.4	35.3
GPT4	53.4	51.6	52.5	47.3	45.7	46.5	38.1	52.3	44.1	32.8	45.0	38.0	42.7	57.5	49.0	36.5	49.2	41.9
OUR Q2.5 7B	75.6	76.7	74.6	73.3	74.4	72.3	51.1	47.2	55.6	48.9	45.3	53.3	61.6	58.6	64.9	57.5	54.8	60.6
OUR Q2.5 14B	76.4	77.2	75.6	73.6	74.3	72.8	51.9	47.7	56.9	49.5	45.5	54.2	62.0	58.1	66.4	56.9	53.4	61.0

Table 10: Sentence-level performance on the SIGHAN13, SIGHAN14 and SIGHAN15 test sets. We adopt the results of previous SOTAs from Qiao et al. (2024). It is worth noting that all these models are trained on the SIGHAN series datasets. 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. ‘Q2.5’ is short for Qwen2.5 series models.

might make the sentence slightly smoother, it is an over-correction since the original sentence was error-free.

E.5 Incorrect Thinking of Reasoning Model May Lead to Wrong Corrections

The reasoning model Deepseek R1 (DeepSeek-AI et al., 2025) shows very impressive performance on tasks like math, code, and science. However, as discussed in the main results, Deepseek R1 achieves lower performance on CSC tasks than its non-reasoning variant Deepseek V3. Although Deepseek R1 achieves higher recall than Deepseek V3, it introduces too many over-corrections.

After analyzing the reasoning process of Deepseek R1, we find that incorrect thinking may lead to wrong corrections, resulting in lower performance than Deepseek V3.

Figures 7 to 10 show thinking examples from Deepseek R1.

The first two examples are correct thinking examples, showing that Deepseek R1 can make reasonable thinking and correctly fix errors based on it. Minor flaws in thinking may not affect the correctness of the correction. For example, in the first example, Deepseek R1 considers “埋” (*mái*) and “理” (*lĭ*) to be similar-sounding, but in fact they are look-alikes rather than sound-alikes. However, this does not affect the correction result.

Input	醋酸圈奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
Reference	醋酸曲安奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
ICL	醋酸曲安奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
ICL-RR	醋酸曲安奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
TfPf	醋酸甲奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
OUR	醋酸曲安奈德尿素乳膏是一种复方的外用软膏,主要含有醋酸曲安奈德和尿素。
Input	无效者不需任何理由和解释凭收据10日内免费凋方或按余药退款。
Reference	无效者不需任何理由和解释凭收据10日内免费调方或按余药退款。
ICL	无效者不需任何理由和解释凭收据10日内免费调方或按余药退款。
ICL-RR	无效者不需任何理由和解释凭收据10日内免费调方或按余药退款。
TfPf	无效者不需任何理由和解释凭收据10日内免费调换或按余药退款。
OUR	无效者不需任何理由和解释凭收据10日内免费调方或按余药退款。
Input	当由于上述原因而试乙方无法履行保证时,以方不承担责任。
Reference	当由于上述原因而使乙方无法履行保证时,乙方不承担责任。
ICL	当由于上述原因而使乙方无法履行保证时,甲方不承担责任。
ICL-RR	当由于上述原因而使乙方无法履行保证时,甲方不承担责任。
TfPf	当由于上述原因而使乙方无法履行保证时,乙方不承担责任。
OUR	当由于上述原因而使乙方无法履行保证时,乙方不承担责任。
Input	当月20座大城市房价同比增长4.9%,创下2012年10月来最小增速
Reference	当月20座大城市房价同比增长4.9%,创下2012年10月来最小增速。
ICL	当月20座大城市房价同比增长4.9%,创下2012年10月来最小增速。
ICL-RR	当月20座大城市房价同比增长4.9%,创下2012年10月以来最小增速。
TfPf	当月20座大城市房价同比增长4.9%,创下2012年10月来最小增速。
OUR	当月20座大城市房价同比增长4.9%,创下2012年10月以来最小增速。

Table 11: Qualitative examples of our approach and the baselines using the Qwen2.5 14B model. Corrections marked in "Blue" are correct, while those in "Red" are incorrect.

Example 1: *Correct Thinking (with a little flaw)*

Input

松原市开展清埋整顿和视范房地产市场大检查

Reference

松原市开展清理整顿和规范房地产市场大检查

The city of Songyuan is conducting a major inspection to clean up, regulate, and standardize the real estate market.

DeepSeek V3 Output

松原市开展清理整顿和规范房地产市场大检查

DeepSeek R1 Think

好的，我现在需要处理用户输入的句子，找出可能的错别字并进行纠正，同时尽量减少对原句的改动，不新增、删除或修改标点符号。首先，我要仔细阅读用户提供的例子，了解他们期望的处理方式。\n

用户给的例子中，比如第一个输入中的"班马王子"被纠正为"斑马王子"，这里"班"和"斑"是同音字，但"斑马"是正确的写法。另一个例子是"自然延生"改为"自然诞生"，这里"延"和"诞"字形相近，但"诞生"是正确的。还有"登录"改为"登陆"，虽然发音相同，但"登陆"在这里是正确的动词。这些例子显示，需要关注同音字、形近字以及特定语境下的正确用词。\n

现在来看当前的输入句子："松原市开展清埋整顿和视范房地产市场大检查"。我需要逐词分析可能的错别字。\n

首先，"清埋"这个词是否正确？正确的应该是"清理"吗？因为"埋"和"理"发音相近，但"清理整顿"是常见的搭配，比如清理市场、整顿秩序。所以这里"清埋"可能是"清理"的错别字。\n

接下来是"视范"，正确的应该是"规范"吗？"视"和"规"在字形上有点相似，但发音不同。"规范房地产市场"是合理的表达，而"视范"这个词组不太常见，可能是"规范"的笔误或形近错误。\n

因此，纠正后的句子应该是："松原市开展清理整顿和规范房地产市场大检查"。这样改动后的句子只替换了两个字，符合用户要求的最小改动原则，并且没有影响标点符号。\n

再检查一遍是否有其他可能的错误。比如"整顿"是否正确？是的，没有问题。"房地产市场"也是正确的。标点符号也没有需要改动的地方。所以最终的输出应该是纠正后的这个句子。\n

Okay, I now need to process the user's input sentence, identify possible typos, and correct them while minimizing changes to the original sentence, without adding, deleting, or modifying punctuation. First, I need to carefully read the examples provided by the user to understand their expected handling method.\n

In the examples given by the user, for instance, the first input "班马王子" was corrected to "斑马王子", where "班" and "斑" are homophones, but "斑马" is the correct spelling. Another example is "自然延生" changed to "自然诞生", where "延" and "诞" have similar shapes, but "诞生" is correct. Also, "登录" changed to "登陆", although they sound the same, "登陆" is the correct verb here. These examples show the need to pay attention to homophones, similar-looking characters, and the correct word usage in specific contexts.\n

Now, let's look at the current input sentence: "松原市开展清埋整顿和视范房地产市场大检查". I need to analyze each word for possible typos.\n

First, is the word "清埋" correct? Should it be "清理"? Because "埋" and "理" sound similar, but "清理整顿" is a common phrase, like cleaning up the market and rectifying order. So "清埋" might be a typo for "清理".\n

Next is "视范", should it be "规范"? "视" and "规" look somewhat similar, but sound different. "规范房地产市场" is a reasonable expression, while "视范" is not common and might be a typo or similar-looking error for "规范".\n

Therefore, the corrected sentence should be: "松原市开展清理整顿和规范房地产市场大检查". This revised sentence only replaces two characters, adhering to the user's principle of minimal changes, and does not affect punctuation.\n

Check again for any other possible errors. For example, is "整顿" correct? Yes, no problem. "房地产市场" is also correct. There is no need to change punctuation. So the final output should be this corrected sentence.\n

DeepSeek R1 Output

松原市开展清理整顿和规范房地产市场大检查

Figure 7: An example of Correct Thinking

Example 2: *Correct Thinking*

Input

目前上暂无任何一种治疗方式可保证高热惊厥不复发。

Reference

目前暂无任何一种治疗方式可保证高热惊厥不复发。

Currently, there is no treatment available that can guarantee the prevention of recurrent febrile seizures.

DeepSeek V3 Output

目前尚暂无任何一种治疗方式可保证高热惊厥不复发。

DeepSeek R1 Think

好的，我现在需要处理用户的输入，找出可能的错别字、多字或漏字，并尽可能少地修改原句。首先，用户提供的例子显示，他们希望保持标点符号不变，只调整文字部分。\n

现在来看当前的输入句子："目前上暂无任何一种治疗方式可保证高热惊厥不复发。"我需要仔细分析每个部分是否有错误。\n

首先，"目前上暂无"这里可能有问题。"目前上"听起来不太自然，正确的表达应该是"目前"或者"目前临床上"。但原句中的"上"是否多余呢？比如，正确的说法可能是"目前暂无"，而"上"可能是多出来的字。例如，用户可能想表达"目前暂无"，但误加了"上"字。因此，"目前上"应该改为"目前"，去掉"上"字。\n

接下来检查其他部分："暂无任何一种治疗方式可保证高热惊厥不复发。"这里的"暂无"是否正确？"暂无"表示暂时没有，符合语境。其他词语如"治疗方式"、"保证"、"复发"都没有明显错误。标点符号也没有问题，句尾是句号，符合要求。\n

所以，修改后的句子应该是："目前暂无任何一种治疗方式可保证高热惊厥不复发。"去除了多余的"上"字，使句子更通顺，同时没有改动其他部分，符合用户的要求。\n

Alright, I now need to process the user's input, identify possible typos, extra words, or missing words, and make minimal changes to the original sentence. First, the examples provided by the user show that they want to keep the punctuation unchanged and only adjust the text part.\n

Now, let's look at the current input sentence: "目前上暂无任何一种治疗方式可保证高热惊厥不复发." I need to carefully analyze each part for errors.\n

First, "目前上暂无" might have an issue. "目前上" doesn't sound natural; the correct expression should be "目前" or "目前临床上". But is the "上" in the original sentence redundant? For example, the correct phrase might be "目前暂无", and "上" might be an extra word. For example, the user might have intended to say "目前暂无" but mistakenly added "上". Therefore, "目前上" should be changed to "目前", removing the "上".\n

Next, check the other parts: "暂无任何一种治疗方式可保证高热惊厥不复发." Is "暂无" correct here? "暂无" means temporarily unavailable, which fits the context. Other words like "治疗方式", "保证", "复发" have no obvious errors. The punctuation is also correct, with a period at the end, meeting the requirements.\n

So, the revised sentence should be: "目前暂无任何一种治疗方式可保证高热惊厥不复发." The extra "上" is removed, making the sentence smoother, while no other parts are changed, meeting the user's requirements.\n

DeepSeek R1 Output

目前暂无任何一种治疗方式可保证高热惊厥不复发。

Figure 8: Another example of Correct Thinking

Example 2: *Overthinking*

Input

新抚区已清理违法刀匾近70处将扩大整治范围发现一处拆除一处

Reference

新抚区已清理违法刀匾近70处将扩大整治范围发现一处拆除一处

Xinfu District has removed nearly 70 illegal protruding vertical signboards and will expand the rectification area - removing them as they are discovered.

DeepSeek V3 Output

新抚区已清理违法刀匾近70处将扩大整治范围发现一处拆除一处

DeepSeek R1 Think

好的，我现在需要处理用户的输入，找出可能的错别字并进行纠正，同时尽量少改动原句，不增删或修改标点。首先，我需要仔细阅读用户提供的例子，理解他们的处理方式和规则。\n

用户的输入是："新抚区已清理违法刀匾近70处将扩大整治范围发现一处拆除一处"。我要逐词分析，看看有没有可能的错别字。首先，"刀匾"这个词听起来不太对，可能是一个错误。通常我们会说"刀牌"或者"招牌"，但"刀匾"似乎不太常见。可能用户是想表达"刀牌"，而"匾"可能是一个错别字，正确的应该是"刀牌"或者"招牌"。比如，违法设置的广告牌、招牌等，所以这里"刀匾"可能是"刀牌"的误写，或者"招牌"的错误。但"刀牌"也不太常见，可能是"招牌"中的"招"被误写为"刀"。因为"招"和"刀"在字形上有点相似，尤其是在手写或快速输入时容易出错。所以可能的正确词是"招牌"，即"违法招牌"。\n

接下来，验证这个推测是否符合语境。原文在说清理违法设置的物品，通常这类整治行动针对的是违规广告牌、招牌等，所以"招牌"更符合语境。因此，"刀匾"应纠正为"招牌"。这样修改后，句子变为："新抚区已清理违法招牌近70处将扩大整治范围发现一处拆除一处"。检查其他部分是否有错误，比如"近70处将扩大"这里的"将"是否正确，是的，没有问题。其他部分也没有明显的错别字，所以最终的纠正应该是将"刀匾"改为"招牌"。\n

另外，需要注意标点符号是否被改动，原句中没有标点，所以输出也应该保持无标点，但根据例子中的情况，用户可能允许保持原标点结构。不过在这个输入中，原句没有使用标点，所以输出也不需要添加。但可能用户希望保持原样，所以不需要添加标点。因此，最终输出应为："新抚区已清理违法招牌近70处将扩大整治范围发现一处拆除一处"。\n

Alright, I now need to process the user's input, identify possible typos, and make corrections while minimizing changes to the original sentence, without adding, deleting, or modifying punctuation. First, I need to carefully read the examples provided by the user to understand their processing methods and rules.\n

The user's input is: "新抚区已清理违法刀匾近70处将扩大整治范围发现一处拆除一处". I need to analyze word by word to see if there are any possible typos. First, the word "刀匾" doesn't sound quite right and might be an error. We usually say "刀牌" or "招牌", but "刀匾" seems uncommon. The user might mean "刀牌", and "匾" might be a typo, with the correct word being "刀牌" or "招牌". For example, illegal advertising signs, signboards, etc., so here "刀匾" might be a miswriting of "刀牌" or an error for "招牌". However, "刀牌" is also uncommon, and it might be that "招" in "招牌" was mistakenly written as "刀". Since "招" and "刀" are somewhat similar in shape, especially in handwriting or fast input, they are easy to confuse. Therefore, the correct word might be "招牌", meaning "illegal signboards".\n

Next, verify if this hypothesis fits the context. The original text talks about clearing illegally set items, and such rectification actions usually target illegal advertising signs, signboards, etc., so "招牌" fits the context better. Therefore, "刀匾" should be corrected to "招牌". After this modification, the sentence becomes: "新抚区已清理违法招牌近70处将扩大整治范围发现一处拆除一处". Check if there are errors in other parts, such as whether "将" in "近70处将扩大" is correct. Yes, there is no problem. There are no obvious typos in other parts, so the final correction should be changing "刀匾" to "招牌".\n

Additionally, pay attention to whether punctuation has been altered. The original sentence has no punctuation, so the output should also remain without punctuation. However, based on the examples, the user might allow maintaining the original punctuation structure. But in this input, the original sentence does not use punctuation, so the output does not need to add any. But, the user might want to keep it as is, so no punctuation needs to be added. Therefore, the final output should be: "新抚区已清理违法招牌近70处将扩大整治范围发现一处拆除一处".\n

DeepSeek R1 Output

新抚区已清理违法招牌近70处将扩大整治范围发现一处拆除一处

Figure 9: An example of Incorrect Thinking

Example 3: *Oversight*

Input

行窃是触电身亡,小偷家属索赔,赔还是不赔

Reference

行窃时触电身亡,小偷家属索赔,赔还是不赔

Died of electrocution while stealing, the thief's family claims compensation—compensate or not.

DeepSeek V3 Output

行窃时触电身亡,小偷家属索赔,赔还是不赔

DeepSeek R1 Think

好，我现在需要处理用户输入的句子，找出可能的错别字并进行纠正，同时尽量少改动原句。首先，我需要仔细阅读输入的句子：\n

输入: 行窃是触电身亡,小偷家属索赔,赔还是不赔\n

首先，逐词检查有没有明显的错别字。第一个部分"行窃是触电身亡"，看起来没问题，"行窃"指的是偷窃行为，"触电身亡"正确。接着是"小偷家属索赔"，这里"家属"正确，没有错别字。最后是"赔还是不赔"，这里"赔"字重复使用，但结构正确，没有问题。\n

再检查标点符号是否正确。用户使用了逗号分隔各部分，没有使用错误，比如句号或分号，所以标点符号不需要改动。\n

接下来，考虑是否有同音字或形近字错误。例如，"触电"是否应为"触电"？是的，正确。"身亡"正确。其他词汇如"行窃"、"家属"、"索赔"都正确。\n

有没有可能被忽略的错误？比如"行窃"是否应为"行窃"？不，是正确的。"触电"是否有其他写法？比如"电击"，但原句用"触电"更准确，指接触电流导致死亡，正确。\n

所以整个句子看起来没有错别字，结构和用词都正确。因此，输出应该与输入相同，不需要修改。\n

Ok, I now need to process the user's input sentence, identify possible typos, and make corrections while minimizing changes to the original sentence. First, I need to carefully read the input sentence:\n

Input: 行窃是触电身亡,小偷家属索赔,赔还是不赔\n

First, check word by word for any obvious typos. The first part "行窃是触电身亡" seems fine, "行窃" refers to the act of stealing, "触电身亡" is correct. Next is "小偷家属索赔", here "家属" is correct, no typos. Finally, "赔还是不赔", the word "赔" is repeated, but the structure is correct, no issues.\n

Then check if the punctuation is correct. The user used commas to separate each part, no errors like periods or semicolons, so the punctuation does not need to be changed.\n

Next, consider if there are homophones or similar-looking character errors. For example, should "触电" be "触电"? Yes, it's correct. "身亡" is correct. Other words like "行窃", "家属", "索赔" are all correct.\n

Are there any errors that might have been overlooked? For example, should "行窃" be "行窃"? No, it's correct. Is there another way to write "触电"? Like "电击", but the original sentence uses "触电" more accurately, referring to death by electric shock, correct.\n

So the whole sentence seems to have no typos, the structure and wording are correct. Therefore, the output should be the same as the input, no modifications needed.\n

DeepSeek R1 Output

行窃是触电身亡,小偷家属索赔,赔还是不赔

Figure 10: Another example of Incorrect Thinking

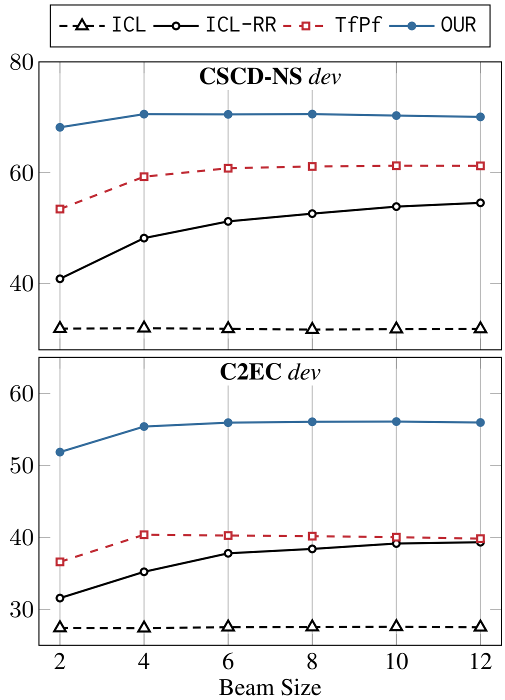


Figure 11: Results of different beam sizes.

In the third example, "刀匾" (*vertical signboards*, *dāo biǎn*) is an uncommon but correct word. However, Deepseek R1 over-thinks about it and incorrectly changes it to "招牌" (*signboard*, *zhāo pái*), a more common word with a similar but slightly different meaning.

In the fourth example, Deepseek R1 analyzes all words in the input sentence but fails to consider whether the "是" in the input sentence is correct or not. This oversight leads to an under-correction.

Nevertheless, we believe that the potential of the reasoning model Deepseek R1 is still huge. We plan to further investigate the use of reasoning models in CSC and C2EC tasks in the future.

F More Discussions

F.1 Impact of Beam Size

Figure 11 shows the performance of our method with varying beam sizes on the CSCD-NS and C2EC datasets. The results indicate that our method performs well even with a small beam size. In particular, a beam size of 2 is sufficient to surpass ICL, ICL-RR, and TfPf on both datasets. Interestingly, as the beam size increases, the performance of ICL remains almost unchanged, while the performance of ICL-RR steadily improves. This suggests that while ICL can find a set of good candidates, it struggles to rank them properly.

System	CSCD-NS *dev*			C2EC *dev*		
	P	R	F_1	P	R	F_1
OUR	67.70	73.69	70.57	65.14	49.18	56.05
-FR	62.28	76.16	68.52	60.34	54.74	57.40
-LR	71.25	68.13	69.66	64.51	45.36	53.26

Table 12: Ablation study on the impact of two rewards.

F.2 Impact of Two Rewards

In Equation 4, we adopt both the faithfulness reward and the length reward from TfPf into the final scoring function. This section investigates the effectiveness of these two rewards. The ablation results are shown in Table 12.

When the faithfulness reward is removed, our method shows an increase in recall, while the precision is reduced. Conversely, removing the length reward results in better precision, but this comes with a decline in recall.

The faithfulness reward mainly improves precision, while the length reward mainly improves recall. Together, they complement each other, leading to better overall performance.

F.3 Adjusting the Precision-Recall Trade-off

In real-world applications, depending on user needs, higher recall may be preferred over precision, or vice versa. For example, newspaper editors, who can verify the correctness of model-generated corrections, might prefer higher recall to identify as many potential errors as possible.

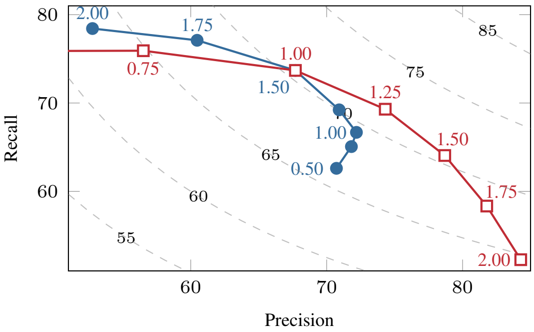
Our method offers two ways to adjust the precision-recall trade-off: 1) Adjusting the temperature of the prompt-based LLM. 2) Introducing a new parameter, γ, as a coefficient for the distance metric term, modifying Equation 2 as follows:

$$\begin{aligned} s(\boldsymbol{x}, \boldsymbol{y}) = &\log p_{\texttt{LLM}}(\boldsymbol{y} \mid \texttt{Prompt}(\boldsymbol{x})) \\ &+ \log p_{\texttt{LLM}}(\boldsymbol{y}) - \gamma\, \texttt{Dist}_{\texttt{L}}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \tag{14}$$

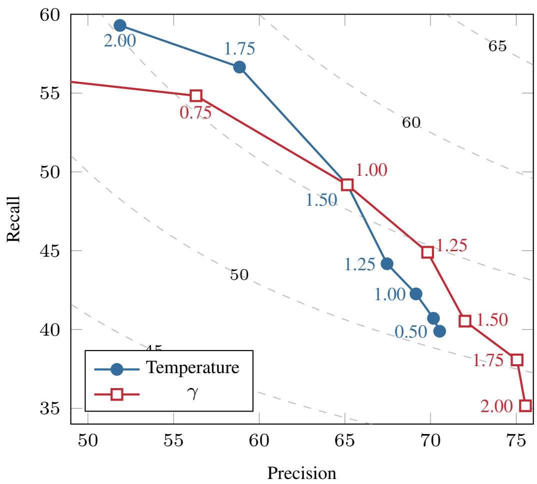
Figure 12 illustrates the precision-recall trade-off curve with F1 score iso-contours.

As observed in Figure 12, for temperature values between 1.0 and 2.0, increasing the temperature leads to higher recall but lower precision. On the CSCD-NS dataset, using a temperature value below 1.0 results in a decrease in both precision and recall.

The γ parameter, which controls the influence of the distance metric, has an opposite effect on the precision-recall trade-off. A higher γ value results in higher precision but lower recall.



(a) **CSCD-NS** *dev*



(b) **C2EC** *dev*

Figure 12: Precision-Recall trade-off curve with F1 score iso-contours. The curve shows the relationship between precision and recall, while the dashed lines represent constant F1 score values. The values near the nodes are the temperature values. In main experiments, we set the temperature to 1.50 and γ to 1.00.

The figure indicates that the γ parameter is more effective for increasing precision, whereas the temperature parameter is more effective for increasing recall.

F.4 Run-time Analysis

We randomly sampled 50 sentences of varying lengths from the development set of CSCD-NS and C2EC to evaluate the running time of our method on `Qwen2.5 7B` compared to the baselines. The experiment was conducted on a single NVIDIA A100 40GB GPU with the Intel Xeon Gold 6248R (3.00GHz) CPU. The batch size was set to 1 during evaluation. The results, as shown in Table 13, indicate that our approach is approximately twice slower than `TfPf`. This increased time is due to the two forward passes of the large language model at each step to obtain the final score. However, since these two forward passes are independent, the process can be accelerated by parallelizing them if more GPUs are available.

System	*per* **Sent.**	*per* **Char.**	**Mem**
Length: < 32			
`ICL (Greedy)`	703.3	30.5	15650
`ICL`	1379.2	59.8	20025
`TfPf`	1118.3	48.5	16404
`Our`	1944.5	84.3	16670
Length: $32 \sim 64$			
`ICL (Greedy)`	1322.4	28.3	15677
`ICL`	1957.9	41.9	20234
`TfPf`	1951.4	41.8	16411
`Our`	3715.1	79.6	16792
Length: > 64			
`ICL (Greedy)`	2182.9	26.7	15684
`ICL`	2894.8	35.4	20297
`TfPf`	3222.5	39.4	16421
`Our`	6155.7	75.3	16956

Table 13: Runtime and memory usage comparison between baseline and our method. The unit of time is ms, and the unit of memory is MB.

CSC In-Context Prompt for Baseline

System Prompt:
你是一个优秀的中文纠错模型，中文纠错模型即更正用户输入句子中的错误。\n
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，在纠正错别字的同时尽可能减少对原句子的改动(不新增、删除和修改标点符号)。只输出没有错误的句子，不要添加任何其他解释或说明。如果句子没有错误，就直接输出和输入相同的句子。\n\n
<Example>\n
输入：{INPUT_EXAMPLE_1: x_1}\n
输出：{OUTPUT_EXAMPLE_1: y_1}\n\n
输入：{INPUT_EXAMPLE_2: x_2}\n
输出：{OUTPUT_EXAMPLE_2: y_2}\n\n
</Example>\n\n

输入：{INPUT: x}\n输出：

System Prompt:
You are an excellent Chinese error correction model, which means you correct errors in the sentences entered by the user.\n
User Prompt:
You need to identify and correct possible misspelled characters in the user's input sentence, and output the correct sentence. While making corrections, try to minimize changes to the original sentence (without adding, deleting, or modifying punctuation). Only output the corrected sentence; do not add any further explanations or notes. If the sentence is error-free, output the exact same sentence as the input.\n
<Example>\n
Input: {INPUT_EXAMPLE_1: x_1}\n
Output: {OUTPUT_EXAMPLE_1: y_1}\n\n
Input: {INPUT_EXAMPLE_2: x_2}\n
Output: {OUTPUT_EXAMPLE_2: y_2}\n\n
</Example>\n\n

Input: {INPUT: x}\n
Output:

C2EC In-Context Prompt for Baseline

System Prompt:
你是一个优秀的中文纠错模型，中文纠错模型即更正用户输入句子中的错误。\n
User Prompt:
你需要识别并纠正用户输入的句子中可能的错别字、多字、漏字并输出正确的句子，在修改的同时尽可能减少对原句子的改动(不新增、删除和修改标点符号)。只输出没有错误的句子，不要添加任何其他解释或说明。如果句子没有错误，就直接输出和输入相同的句子。\n\n
<Example>\n
输入：{INPUT_EXAMPLE_1: x_1}\n
输出：{OUTPUT_EXAMPLE_1: y_1}\n\n
输入：{INPUT_EXAMPLE_2: x_2}\n
输出：{OUTPUT_EXAMPLE_2: y_2}\n\n
</Example>\n\n

输入：{INPUT: x}\n输出：

System Prompt:
You are an excellent Chinese error correction model, which means you correct errors in the sentences entered by the user.\n
User Prompt:
You need to identify and correct possible misspellings, redundant and missing characters in the user's input sentence, and output the correct sentence. While making corrections, try to minimize changes to the original sentence (without adding, deleting, or modifying punctuation). Only output the corrected sentence; do not add any further explanations or notes. If the sentence is error-free, output the exact same sentence as the input.\n
<Example>\n
Input: {INPUT_EXAMPLE_1: x_1}\n
Output: {OUTPUT_EXAMPLE_1: y_1}\n\n
Input: {INPUT_EXAMPLE_2: x_2}\n
Output: {OUTPUT_EXAMPLE_2: y_2}\n\n
</Example>\n\n

Input: {INPUT: x}\n
Output:

Figure 13: In-context learning prompts for baseline models.