Towards More Realistic Chinese Spell Checking with New Benchmark and Specialized Expert Model

Yue Wang[1,2*], **Zilong Zheng**[1*], **Juntao Li**[1†], **Zhihui Liu**[2],
Jinxiong Chang[2], **Qishen Zhang**[2], **Zhongyi Liu**[2], **Guannan Zhang**[2], **Min Zhang**[1]
[1]School of Computer Science and Technology, Soochow University
[2]Ant Group
ywangnlp@stu.suda.edu.cn

Abstract

Large Language Models (LLMs) hold considerable promise for artificial general intelligence, given their intrinsic abilities to accomplish a wide range of open-domain tasks either independently or in tandem with specialized expert models. However, despite these capabilities, the performance of LLMs has yet to be comprehensively evaluated in realistic scenarios. To this end, in this work, we introduce a novel task, the **R**ealistic **C**hinese **S**pell **C**hecking (**RCSC**), to evaluate the effectiveness of existing methods comprehensively. In contrast to existing works that solely address Chinese character misspellings or pinyin conversions, our task aims to convert the realistic Chinese text into the corresponding correct text. The realistic Chinese text may potentially contain both Chinese misspellings and pinyin conversions. We first present the **R**ealistic **C**hinese **S**pell **C**hecking **B**enchmark (**RCSCB**), which consists of two subsets and contains a total of 581,657 samples. Then, we benchmark the performance of various baselines and find that all the existing methods, including instruction-based LLMs, achieve unsatisfactory results on RCSCB. To further improve the performance on RCSCB, we propose **P**inyin-**E**nhanced **S**pell **C**hecker (**PESC**), which is specifically designed to address pinyin-related misspellings. Experimental results demonstrate that PESC can achieve state-of-the-art performance on RCSCB. Despite the progress made, the current state-of-the-art performance is still far from satisfactory. We expect further progress on this crucial and challenging task. Our code and dataset are available at GitHub https://github.com/AlipaySEQ/PESC.

Keywords: chinese spelling check, pinyin input method

1. Introduction

Recently, Large Language Models (LLMs) have gained global attention due to their remarkable ability to complete a wide range of tasks based on user instructions, either independently or by connecting with specialized expert models, e.g., Hugginggpt (Shen et al., 2023) and Taskmatrix.ai (Liang et al., 2023). With the potential to serve as artificial general intelligence, the research community has focused on evaluating the performance of LLMs across various Natural Language Processing (NLP) tasks (Wei et al., 2023; Hu et al., 2023; Jiao et al., 2023; Fang et al., 2023; Wu et al., 2023). The results indicate that LLMs can achieve competitive results in diverse NLP tasks, such as information extraction (Wei et al., 2023; Hu et al., 2023), text summarization (Wang et al., 2023) and machine translation (Jiao et al., 2023). However, in the Grammatical Error Correction (GEC) task, LLMs exhibit poor performance in terms of Precision and $F_{0.5}$ score due to the limitation of over-corrections (Fang et al., 2023; Wu et al., 2023). In this work, following the line of these works, we explore the performance in more realistic scenarios.

Chinese	Pinyin	CM	PC	Chinese Text	Existing Works	Ours
✓				今天天气真不错 The weather is really nice today	✓	✓
✓		✓		今天天气镇(town)不错	✓	✓
	✓			jin tian tian qi zhen bu cuo 今天天气真不错	✓	✓
	✓		✓	jin tian tia qi zhen bu cuo	✓	✓
✓	✓			今天tian气zhen不错		✓
✓	✓	✓	✓	今天tia气镇不错		✓

Table 1: The misspelling types of Chinese text in realistic scenarios, where **CM** and **PC** denote Chinese misspellings and pinyin conversions, respectively. All the misspellings are marked in red. Existing works focus on addressing the first four misspelling types, while this work encompasses all types of misspellings.

The pinyin input method is the most popular for Chinese users to type Chinese characters on electronic devices, which needs a two-stage process [1]. Specifically, users first input pinyin characters, and the pinyin input method provides candidate Chinese characters based on the entered pinyin characters. Subsequently, the users select the intended Chinese characters. While typing Chinese characters, due to the possibility of pressing the wrong key,

* Equal contribution. Work is done during the internship of Yue Wang at Ant Group.
† Corresponding author

LREC-COLING 2024, pages 16570–16580

users may input pinyin characters directly without converting them into intended Chinese characters, mixing pinyin and Chinese characters [2,3]. In formal scenarios, users can check the input content and correct misspellings. However, in specific informal scenarios, such as search engine queries and instant messaging applications, users input texts casually and may need to transmit messages without meticulous inspection, contributing to the misspellings of mixed pinyin and Chinese characters and hindering the understanding of the original intention, as shown in Table 1. However, existing works primarily focus on either correcting incorrect Chinese characters (Chinese Spell Checking) or converting Pinyin characters to Chinese characters (Pinyin Input Method), leaving this problem under-explored.

To fill this gap, in this work, we introduce the **R**ealistic **C**hinese **S**pell **C**hecking (**RCSC**) task, which aims to correct both Chinese character and pinyin misspellings. We first construct the Realistic Chinese Spell Checking Benchmark (RCSCB), which is collected from two existing benchmarks and has undergone meticulous and comprehensive data processing. Then, we test the performance of various baselines on RCSCB, including instruction-based large language models (LLMs), fine-tuned Seq2Seq pre-trained language models, and Chinese Spell Checking methods. Surprisingly, instruction-based LLMs struggle with this task due to over-correction problems. Finally, to achieve a specified expert model to address pinyin-related misspellings, we propose **P**inyin-**E**nhanced **S**pell **C**hecker (**PESC**). Specifically, we introduce a pinyin detector to avoid over-correction on English characters and a pinyin segmenter to ensure that consecutive pinyin characters belonging to different Chinese characters are not misinterpreted as a single character. Besides, we also introduce glyph and phonetic information embedding to enhance the representation capability. Experimental results confirm the effectiveness of PESC, which can achieve state-of-the-art performance on RCSCB with relatively low computation cost.

In a nutshell, our contributions are as follows:

- We propose the **R**ealistic **C**hinese **S**pell **C**hecking (**RCSC**) task. To the best of our knowledge, we are the first to investigate Chinese Spell Checking with both Chinese misspellings and pinyin errors.
- We introduce the benchmark RCSCB and test the performance of various baselines. None of the existing methods achieve satisfactory performance, highlighting the research value of our proposed task.
- We proposed PESC, a specialized model designed to handle pinyin-related misspellings. PESC achieves state-of-the-art performance on the RCSCB benchmark while maintaining relatively low computational overhead.

2. Related Work

2.1. Chinese Spell Checking

Chinese Spell Checking (CSC) aims to detect and correct spelling errors. Early works apply rule-based and statistical-based methods to CSC tasks (Yu and Li, 2014; Chang et al., 2015). With the popularity of the Pre-trained Language Models(PLMs) (Devlin et al., 2019; Liu et al., 2019), recent works wildly use BERT (Devlin et al., 2019) as a backbone model and achieve great progresses in CSC. Hong et al. (2019) use a BERT-based denoising autoencoder to generate candidates and a Viterbi algorithm to select the best ones. Zhang et al. (2020) connect an error detection modular for error correction based on BERT. Guo et al. (2021) propose a global attention decoder to model the relationship between correct and misspelled characters. Li et al. (2021) help the model address the unseen example better by continuously generating pseudo examples. Wang et al. (2021) use an attention-based network to model the relationships between two adjacent characters and propose a pinyin-enhanced candidate generator to generate candidates. Liu et al. (2022) propose a noise modeling module to address multi-typo problems. Li et al. (2022) propose an error-driven contrastive probability optimization framework to prevent the model from predicting common characters. Besides, due to the uniqueness of Chinese characters, some works try to incorporate phonetic and glyph information into PLMs by means of confusion character substitution pre-training strategy (Zhang et al., 2021; Liu et al., 2021), graph convolutional network (Cheng et al., 2020; Ji et al., 2021) and glyph image encoder (Xu et al., 2021). In this work, we focus on a more realistic CSC that aims to correct both Chinese misspellings and pinyin conversions. To better handle this problem, we introduce a benchmark to test performance and propose some strategies to improve the performance of existing methods.

2.2. Pinyin Input Method

The pinyin input method converts the Pinyin sequences of user inputs into corresponding Chinese characters, e.g., "jin tian" for "今天(today)". Due to its practical value in realistic scenarios, there has been a lot of work focused on it. Chen and Lee (2000) propose a trigram language model and a statistically based segmentation. Jia and Zhao (2014) propose a joint graph model to globally optimize pinyin-to-Chinese (P2C) conversion and pinyin typo

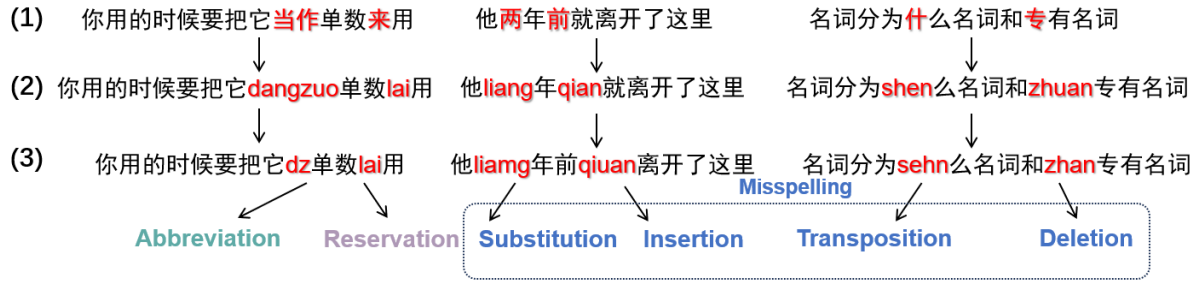

Figure 1: The illustration of pinyin conversion process. In line (1), the characters marked in red represent the Chinese characters selected to convert into pinyin; In line (2), they are converted into the corresponding perfect pinyin; In line (3), they are conducted with pinyin conversions. The pinyin conversion encompasses **Abbreviations**, **Reservations**, and **Misspellings**. The **Misspellings** include **Substitution**, **Insertion**, **Transposition**, and **Deletion**.

correction. Huang et al. (2018) combine attention-based neural machine translation model and information retrieval. Zhang et al. (2019) propose a neural P2C conversion model with open vocabulary learning. Tan et al. (2022) explore adapting pre-trained Chinese GPT to the pinyin input method. In this work, different from these works that focus on how to convert pinyin sequences to Chinese word sequences better, we explore a more realistic scenario in that the input text contains both Chinese misspellings and pinyin conversions.

Dataset	Sen	Len	CM	PC
SIGHAN & Wang271K (Train)	284,201	45.7	398,749	649,427
TAL (Train)	284,136	27.3	220,823	354,125
SIGHAN15 (Test)	1100	32.7	703	1,754
TAL (Test)	12,220	27.1	9,527	15,072

Table 2: The statistics of used datasets, where **Sen** denotes the number of sentences, **Len** denotes the average length of the sentences in each dataset. **CM** and **PC** denote the number of Chinese misspellings and pinyin conversions, respectively.

3. Realistic Chinese Spell Checking

3.1. Task Formulation

In our Realistic Chinese Spell Checking (RCSC) task, considering a text sequence $X = \{x_1, x_2, x_3, x_4, \cdots, x_n\}$ consisting of n characters, wherein x_i denotes a character of Chinese, English, or pinyin, the goal of RCSC is to convert the input text X into its corresponding correct text sequence of m characters $Y = \{y_1, y_2, \cdots, y_m\}$. Because one Chinese character usually corresponds to multiple pinyin characters, this one-to-many relationship causes that m is smaller than n. To complete this goal, the models need to convert pinyin characters into corresponding Chinese characters and correct the Chinese and pinyin misspellings. Besides, in this task, we do not introduce English character misspellings, and hence the models also need to keep all the English characters without modifying them.

3.2. Realistic Chinese Spell Checking Benchmark

Due to the lack of open-source datasets that satisfy the requirements of RCSC, we first introduce the Realistic Chinese Spell Checking Benchmark through meticulous and comprehensive data processing. We collect the original texts from two sources: Tomorrow Advancing Life English lecture audio (TAL)[1] and SIGHAN benchmark (Wu et al., 2013; Yu et al., 2014; Chu and Lin, 2015; Wang et al., 2018). Because realistic Chinese texts often contain both Chinese and English characters, to meet this requirement, we first collect the texts from TAL, which is converted from English lecture audio in China and hence are mixed Chinese and English. Besides, SIGHAN is the most popular benchmark for CSC, whose Chinese character misspellings are collected from realistic scenarios. Although we collect texts from realistic scenarios, both of them cannot contain all misspelling types in RCSC. For TAL, we replace the correct Chinese character with the use of a confusion set (Wang et al., 2019). The replacement probability is set to 4%, which is close to the error rate of SIGHAN. Besides, because the texts of both sources do not contain pinyin, we add pinyin conversion in two raw datasets. Specifically, for Chinese characters, we first randomly choose them with a probability of 6%. Then, for each selected Chinese character, we use the open-source tool pypinyin[2] to convert it into perfect pinyin. Next, we make the following operations

[1] https://ai.100tal.com/openData/voice
[2] https://github.com/mozillazg/python-pinyin

Input	BERT Output	Target
disturb事打扰bu要打扰伽	distur不是打扰不要打扰他	disturb是打扰不要打扰他
mind思xiayng主意	名思想主意	mind思想主意
这里要符合第一个yuanzc前面肯定后免否定人称	这里要符合第一个原则词前面肯定后面否定人称	这里要符合第一个原则前面肯定后面否定人称
翻译做第eureg	翻译做第二二个	翻译做第二个
无liq闹的无关的信息	无聊闹的无关的信息	无理取闹的无关的信息
这里应该yyi哥动词对吧	这里应该一个动词对吧	这里应该用一个动词对吧

Table 3: Examples of input, the corresponding output of BERT, and the target. Chinese misspellings or pinyin conversions, incorrect modifications, and golden modifications are marked in red, orange, and blue respectively.

on the perfect pinyin with the same probability: (a) reserving (reservation); (b) keeping only the first pinyin character (abbreviation); (c) replacing with a pinyin misspelling (misspelling). Following Chen and Lee (2000), we introduce four types for the pinyin misspellings: (a) replacing a pinyin character randomly (substitution); (b) inserting a pinyin character randomly (insertion); (c) deleting a pinyin character randomly (deletion); (d) selecting two consecutive pinyin characters randomly and swapping their order (transposition), as shown in Figure 1. The probabilities of the four types are set to 40%, 20%, 20%, and 20%, respectively. Finally, we collected 581,657 samples in total. Table 2 shows the data statistics. Besides, to ensure the quality of the collected dataset, we remove sentences that are too short and only contain numbers

We utilize the TAL training set and SIGHAN&Wang271k as separate training sets. The model trained using the TAL training set is evaluated on the TAL test set, while the model trained with SIGHAN13, 14, and 15&Wang271k is evaluated on the SIGHAN test set in 2015 (SIGHAN15).

3.3. Dataset Limitation

Our dataset is collected from lecture audio and Chinese exams, which may contain violence, pornography, and political prejudice. Furthermore, we do not consider the numerous Chinese dialects, and the performance of our proposed method may slightly decline when dealing with different dialects.

3.4. Automatic Metrics

Following existing CSC works (Xu et al., 2021; Li et al., 2022), we use sentence-level detection and correction accuracy, precision, recall, and F1 scores as automatic metrics. Specifically, at the detection level, a sentence is deemed correct only if all misspellings in the sentence are successfully detected. At the correction level, the model must not only detect but also correct all incorrect characters to their respective correct forms.

4. PESC Model

Table 3 shows some examples of input, the corresponding output of BERT, and the target of RCSCB. We can see the following problems: (1) some English characters will be over-corrected. (2) some pinyin characters of a single Chinese character will be predicted as several characters. (3) some consecutive pinyin characters belonging to different Chinese characters will be misinterpreted as a single character. To solve these problems and improve model performance on RCSCB, we introduce the PESC model, which consists of a corrector, detector, and segmenter. Figure 2 illustrates that we incorporate phonetic embedding and graph embedding into the embedding layer. Next, the representations obtained from the Bert-based encoder are used as inputs to the detector, segmenter, and corrector, respectively. The corrector outputs the correction result. Finally, we refine the result with the assistance of the detector and segmenter.

4.1. Corrector

The corrector is responsible for rectifying Chinese and pinyin misspellings and converting pinyin to the corresponding Chinese characters. Due to the typical correspondence of multiple pinyin characters to a single Chinese character, the lengths of the source and target sequences differ. To address this problem, we expand the target sequence in the training stage. Specifically, we introduce a symbol ϕ in the target sentence at the corresponding positions of all non-initial pinyin characters in the source sequence. We also try other approaches to align source and target sequence, which will be discussed in Section 5.5. Then, the result of correction is $Y = \{y_1, y_2, \cdots, y_n\}$.The prediction probability of the token x_i is defined as:

$$p_i = P_c(y_i = V_j|X) = Softmax(f_c(T(E)))$$

where $P_c(y_i = V_j|X)$ denotes the conditional probability that the token x_i is predicted as the j-th character in the vocabulary V, $E = (e_1, e_2, \cdots, e_n)$ denotes the embedding of X, f_d is a fully-connected

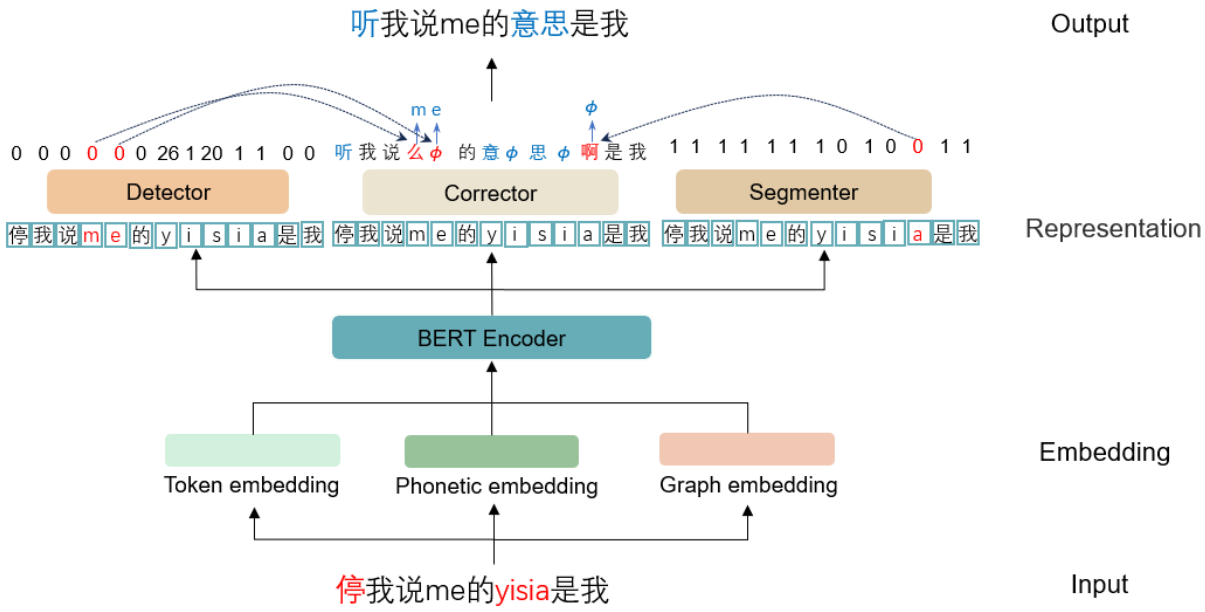

Figure 2: Overview of our model architecture. Phonetic embedding and graph embedding are used to capture phonetic and graph information. Token embedding includes segment embedding and position embedding, which is the same as BERT (Devlin et al., 2019). For example, the input sequence is "停我说me的yisia是我". At the positions of 'm' and 'e',the corrector output are "么" and ϕ. However, the output of the detector at the same positions are all 0, which denotes that 'm' and 'e' are not pinyin characters. We abandon modifying the corrector and keep the original input "me". Meanwhile, the output of the segmenter at the position of 'a' is 0. We change the output of the corrector at the same place to ϕ. Finally, after deleting ϕ, the output is "听我说me的意思是我" (Listen to me, the meaning of 'me' is me.).

layer with an output dimension of length $|V|$, T is a Transformer-based encoder. We model the objective for the training of the correct as follows:

$$\mathcal{L}_c = -\sum_{i=1}^{n} \log P_c(y_i = V_j|X)$$

4.2. Pinyin Detector

The objective of the pinyin detector is to determine whether a character in the input sequence is a pinyin character and to predict the ground-truth character of the initial consonant. We employ a classifier featuring 28 distinct labels. Here, 0 signifies non-relevant characters, while 1 represents pinyin characters that do not serve as initial consonants. To denote the initial consonants, we employ the numerical values 2 through 27 to correspond with the English alphabet letters from a to z, respectively. The output of pinyin detector is a sequence of labels $D = \{d_1, d_2, d_3, \cdots, d_n\}$. For each character, a probability p_i is used to denote the likelihood of being predicted to the ground-truth label, which can be defined as follows:

$$p_i = P_d(d_i = g_i|X) = Softmax(f_d(T(E)))$$

where $P_d(d_i = g_i|X)$ denotes the conditional probability given by the detector, g_i denotes the ground-truth label of x_i, $E = (e_1, e_2, \cdots, e_n)$ denotes the embedding of X, f_d is a fully-connected layer with an output dimension of 28, T is a Transformer-based encoder. The objective for training the pinyin detector is defined as follows:

$$\mathcal{L}_d = -\sum_{i=1}^{n} \log P_d(d_i = g_i|X)$$

In the inference stage, to further utilize the detector for all characters(English and pinyin), if the corresponding positions in the detector's output are 0, we refrain from applying modifications by the corrector and preserve the original input.

4.3. Pinyin Segmenter

The Pinyin segmenter is designed to prevent consecutive pinyin characters belonging to different Chinese characters from being misinterpreted as a single character or the pinyin characters of a single Chinese character from being predicted as several characters. Specifically, we construct a binary classifier in which '0' is assigned to characters that are part of the same Chinese character as their preceding character, while '1' is assigned to other characters. For example, if "今tian" is the input sequence, the label sequence is $\{1, 1, 0, 0, 0\}$, while

"tian" is the pinyin sequence of "天(sky)". The output of pinyin segmenter is a sequence of labels $S = \{s_1, s_2, s_3, \cdots, s_n\}$. For each character, a probability p_i is used to denote the likelihood of being predicted to 1, which can be defined as follows:

$$p_i = P_s(s_i = 1|X) = Softmax(f_s(T(E)))$$

where $P_s(s_i = 1|X)$ denotes the conditional probability given by the segmenter, $E = (e_1, e_2, \cdots, e_n)$ denotes the embedding of X, f_s is a fully-connected layer, T is a Transformer-based encoder. The objective for the training of the pinyin segmenter is defined as follows:

$$\mathcal{L}_s = -\frac{1}{n}\sum_{i=1}^{n}[g_i \log p_i + (1 - g_i)\log(1 - p_i)]$$

where g_i denotes the label of x_i.

In the inference stage, we substitute the output of the corrector with ϕ at positions where the segmenter's output is 0.

4.4. Phonetic and Graph Embedding

Phonetic and graph embedding help the model capture phonetic and graph information, which has been proven effective in many works (Cheng et al., 2020; Xu et al., 2021). Following Rocbert (Su et al., 2022), we define phonetic embedding E_{pho} and graph embedding E_{gra} as follows:

$$E_{pho} = Embedding_p(P)$$

$$E_{gra} = Embedding_g(G)$$

where $Embedding_p$ and $Embedding_g$ denote two embedding functions with different embedding numbers and dimensions, P and G denote phonetic and graph id sequences, respectively, which are initialized with a sequence of zeros matching the length of the input. Next, we perform concatenation in the hidden layer dimension, combining the phonetic embedding E_{pho}, graph embedding E_{gra}, and the standard token embedding E_{tok}, resulting in the concatenated embedding E_{cat}. The final embedding $E$is defined as follows:

$$E = Dropout(LayerNorm(\boldsymbol{W}E_{cat} + \boldsymbol{b}))$$

where $\boldsymbol{W} \in \boldsymbol{R}^{hidden \times em\_dim}$ and $\boldsymbol{b} \in \boldsymbol{R}^{hidden}$ are learnable parameters, $hidden$ is the hidden state size, em_dim is the embedding dimension of E_{cat}.

4.5. Learning and Inference

The learning of our model is conducted end-to-end. The overall objective is to optimize three modules jointly as follows:

$$\mathcal{L} = \lambda_1 L_d + \lambda_2 L_s + (1 - \lambda_1 - \lambda_2)L_c$$

where λ_1 and $\lambda_2 \in$ [0,1] are trade-off coefficients.

In the inference stage, we remove all ϕ in the output. Besides, we propose a pinyin-enhanced decoding strategy. Specifically, we constructed dictionaries using each pinyin letter as a keyword. The dictionaries comprise ϕ and Chinese characters where the initial consonant corresponds to the respective keyword. When decoding pinyin characters, we retain the probabilities of the characters in the corresponding dictionaries and choose the character with the highest probability as the prediction.

5. Experiments

5.1. Baselines

We compare our method with the following baselines: **T5** (Raffel et al., 2020) uses a unified framework for text-to-text transfer learning. **Fastcorrect** (Leng et al., 2021) applies length prediction to enable variable-length correction. **PhVEC** (Fang et al., 2022) utilizes phonological tokens to expand the source sentence, facilitating variable-length correction. **Soft-Masked BERT** (Zhang et al., 2020) comprises a detection network and a correction network. **BERT** (Devlin et al., 2019) directly fine-tune BERT on our datasets. **REALISE** (Xu et al., 2021) captures and combines semantic, pronunciation, and glyph information. **ECOPO** (Li et al., 2022) proposes an error-driven contrastive probability optimization framework to prevent the model from predicting common characters as misspelled ones.

We also compare the following popular LLMs: **ChatGPT** (OpenAI, 2023) is a LLM trained with Reinforcement Learning from Human Feedback (RLHF) approach [3]. **Yuan 1.0** (Wu et al., 2021) is a Chinese LLM with 12B parameters, pretrained on 5TB data [4]. **ChatGLM** (Zeng et al., 2022) is an open bilingual 6.2B LLM based on General Language Model (GLM) framework. **BELLE** (Yunjie Ji et al., 2023) is an instruction tuning LLM with 7B parameters based on Bloom (Scao et al., 2022) [5].We employ few-shot testing to assess the performance of these LLMs. Table 4 displays the specific prompts utilized in our evaluation.

5.2. Experimental Details

We implement our model with the open-sourced Transformers library [6] (Wolf et al., 2020). We ini-

[3] In particular, we use *text-davinci-003.*

[4] The model API is obtained from https://air.inspur.com.

[5] The checkpoint is "bloom7b-2m-8bit-128g"https://huggingface.co/BelleGroup/BELLE-7B-gptq.

[6] https://github.com/huggingface/transformers

Id	Prompt
1	一些句子会含有中文拼音，请按照示例修改给出正确的中文句子。 原句：some肯定有some对吧这个大常见了还有all就sht表示所有hamy。纠正：some肯定有some对吧这个太常见了还有all就是表示所有还有。 原句：rfan后今天下课势间也到了这ge就植能留着下次zjiang了。纠正：然后今天下课时间也到了这个就只能留着下次再讲了。 原句：**inputs**。纠正：
2	请按照示例修改掉句子中错别字和拼音，给出正确的句子。 原句：some肯定有some对吧这个大常见了还有all就sht表示所有hamy。纠正：some肯定有some对吧这个太常见了还有all就是表示所有还有。 原句：rfan后今天下课势间也到了这ge就植能留着下次zjiang了。纠正：然后今天下课时间也到了这个就只能留着下次再讲了。 原句：**inputs**。纠正：
3	请按照示例将句子中的错别字和中文拼音替换为正确的汉字。 原句：some肯定有some对吧这个大常见了还有all就sht表示所有hamy。纠正：some肯定有some对吧这个太常见了还有all就是表示所有还有。 原句：rfan后今天下课势间也到了这ge就植能留着下次zjiang了。纠正：然后今天下课时间也到了这个就只能留着下次再讲了。 原句：**inputs**。纠正：
4	请按照示例请修改错误并给出正确的中文句子。 原句：some肯定有some对吧这个大常见了还有all就sht表示所有hamy。纠正：some肯定有some对吧这个太常见了还有all就是表示所有还有。 原句：rfan后今天下课势间也到了这ge就植能留着下次zjiang了。纠正：然后今天下课时间也到了这个就只能留着下次再讲了。 原句：**inputs**。纠正：

Table 4: The different few-shot prompts used to evaluate LLMs.

Method	Parameters	Detection Level				Correction Level			
		Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
T5-base (Raffel et al., 2020)	300M	-	-	-	-	77.89	71.65	71.09	71.37
Fastcorrect (Leng et al., 2021)	99M	85.82	82.15	81.44	81.79	79.25	73.28	72.65	72.96
PhVEC (Fang et al., 2022)	120M	87.86	84.74	84.35	84.55	76.29	69.17	68.85	69.01
Soft-Masked BERT (Zhang et al., 2020)	140M	79.77	74.99	73.36	74.16	69.18	60.47	59.15	59.80
BERT (Devlin et al., 2019)	120M	90.78	88.58	88.04	88.31	83.05	78.15	77.67	77.91
REALISE (Xu et al., 2021)	280M	91.31	89.43	88.67	89.05	84.57	80.31	79.63	79.96
ECOPO (Li et al., 2022)	120M	90.92	88.77	88.19	88.48	83.39	78.62	78.10	78.36
ChatGPT	175B	-	-	-	-	18.73	12.32	15.19	13.60
Yuan 1.0 (Wu et al., 2021)	12B	-	-	-	-	14.00	3.05	3.13	3.09
ChatGLM (Zeng et al., 2022)	6B	-	-	-	-	1.53	0.62	0.82	0.70
BELLE (Yunjie Ji et al., 2023)	7B	-	-	-	-	7.15	2.07	2.26	2.28
PESC(Ours)	130M	**91.85**	**89.97**	**89.43**	**89.70**	**84.92**	**80.62**	**80.13**	**80.37**

Table 5: The performance of PESC and all baseline models on the TAL test set. ***Bold*** indicates the current state-of-the-art performance. The results of LLMs are the average performance of 4 different prompts.

Method	Parameters	Detection Level				Correction Level			
		Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
T5-base (Raffel et al., 2020)	300M	-	-	-	-	58.67	54.46	54.86	54.66
Fastcorrect (Leng et al., 2021)	99M	74.09	71.73	72.33	72.03	67.09	63.88	64.40	64.14
PhVEC (Fang et al., 2022)	120M	69.69	67.17	69.77	68.45	57.42	53.58	55.65	54.59
Soft-Masked BERT (Zhang et al., 2020)	140M	63.82	62.83	60.40	61.59	55.09	50.43	52.87	51.62
BERT (Devlin et al., 2019)	120M	78.27	76.71	77.12	76.91	70.82	68.24	68.52	68.38
REALISE (Xu et al., 2021)	280M	79.91	78.05	79.01	78.53	72.64	69.92	70.78	70.35
ECOPO (Li et al., 2022)	120M	79.18	77.66	77.98	77.82	71.27	68.58	68.93	68.75
ChatGPT	175B	-	-	-	-	20.25	18.07	19.77	18.88
Yuan 1.0 (Wu et al., 2021)	12B	-	-	-	-	7.50	2.81	2.62	2.71
ChatGLM (Zeng et al., 2022)	6B	-	-	-	-	2.27	1.59	1.75	1.66
BELLE (Yunjie Ji et al., 2023)	7B	-	-	-	-	5.74	3.30	3.54	3.41
PESC(Ours)	130M	**81.45**	**79.90**	**80.98**	**80.44**	**73.91**	**71.34**	**72.31**	**71.82**

Table 6: The performance of PESC and all baseline models on SIGHAN15 test set. ***Bold*** indicates the current state-of-the-art performance. The results of LLMs are the average performance of 4 different prompts.

tialize embedding, encoder, and corrector with the weights of "roc-bert-base-zh" [7]. The parameters

[7] https://huggingface.co/weiweishi/roc-bert-base-zh

of the segmenter and detector will be randomly initialized. The hidden state size of all modules is set to 768. The phonetic embedding function has an embedding number of 910 and a dimension of 768, while the graph embedding function has an embedding number of 24,858 and a dimension of 512. Considering that pinyin sequences cannot be divided into subword units like English words, we implement a character tokenizer that tokenizes all characters into independent units. The tokenizer is used in our model and other CSC methods. We set the learning rate as 5e-5, warm-up steps as 10,000, and batch size as 64. The loss trade-off coefficients λ_1 and λ_2 in joint learning are set to 0.3, and the model is trained with the AdamW (Loshchilov and Hutter, 2017) optimizer for 20 epochs.

5.3. Main Results

From the results in Table 5 and Table 6, we can observe that the Seq2Seq model (T5), as well as the models designed for variable-length error correction (Fastcorrect and PhVEC), exhibited some potential in this task, although they still lag behind CSC models (with the exception of Soft-Masked BERT). Among all the baselines, REALISE exhibits the best performance as it incorporates phonetic and glyph information into the model. Specifically, the phonetic information proves effective in correcting pinyin misspellings. However, Soft-Masked BERT demonstrates poor performance because it causes the embedding at the error position converge to the mask embedding. Consequently, this results in the loss of information about the pinyin characters. The performance of the LLMs is highly deficient. One contributing factor is their difficulty in comprehending the meaning of pinyin, particularly when erroneous pinyin is involved. Another issue arises from their tendency to engage in overcorrection. The PESC model performs better than all baselines on two test sets. Specifically, compared with REALISE, PESC achieves 0.65 F1 score improvements at detection-level and 0.41 F1 score improvements at correction-level on the TAL test set. Additionally, it achieves 1.91 F1 score improvements at detection-level and 1.47 score F1 improvements at correction-level on SIGHAN15.

5.4. Ablation Study

To assess the individual contribution of each component in PESC, we take BERT as the backbone and conduct ablation studies using the following configurations: (1) introducing the segmenter, (2) introducing the detector, (3) introducing the detector and using pinyin enhanced decode, (4) introducing the phonetic and graph embedding. Table 7 shows the ablation results evaluated on the TAL test set.

Method	Acc	Pre	Rec	F1
	Detection Level			
BERT	90.78	88.58	88.04	88.31
+Seg.	91.05	88.90	88.35	88.62
+Det.	91.12	89.02	88.43	88.72
+Det.&PED.	91.16	89.10	88.51	88.80
+ Pho.&Gra. E	91.65	89.75	89.17	89.46
PESC	**91.85**	**89.97**	**89.43**	**89.70**
	Correction Level			
BERT	83.05	78.15	77.67	77.91
+Seg.	83.16	78.25	77.76	78.00
+Det.	83.75	79.07	78.55	78.81
+Det.&PED.	83.86	79.12	78.70	78.91
+ Pho.&Gra. E	84.51	80.10	79.58	79.84
PESC	**84.92**	**80.62**	**80.13**	**80.37**

Table 7: Ablation results of the PESC model on TAL test set, where **Seg.** represents segmenter, **Dec.** represents detector, **PED.**represent pinyin-enhanced decoding, **Pho.&Gra. E** represent phonetic and graph embedding. ***Bold*** indicates the best performance.

Method	Acc	Pre	Rec	F1
	Detection Level			
Ours	**91.00**	**88.92**	**88.23**	**88.57**
All	90.57	88.34	87.75	88.04
Merge	90.96	88.82	88.23	88.52
	Correction Level			
Ours	**83.93**	**79.36**	**78.75**	**79.05**
All	82.63	77.62	77.10	77.36
Merge	83.16	78.28	77.76	78.02

Table 8: Performance of PESC (w/o phonetic and graph embedding) on the TAL test set using different alignment strategies. ***Bold*** indicates the best performance.

It is observed that the inclusion of each module enhances the model performance to varying extents. In particular, the detector module assists the model in discerning pinyin characters from other characters, thereby avoiding excessive correction of English characters. The segmenter module equips the model with the capability to segment consecutive pinyin characters associated with distinct Chinese characters. Moreover, pinyin-enhanced decoding enhances the model's reliance on pinyin characters during inference while avoiding the prediction of these common characters (e.g., predicting "今r(日)" to "今天"). Additionally, the phonetic and graph embedding modules enable the model to acquire phonetic and graph information. With all modules working together, PESC achieves the current state-of-the-art performance.

5.5. The Effect of Alignment Strategy

In the PESC model, we align the source and target sequences by associating the initial consonant with the correct Chinese character and introduc-

ing the character ϕ. Additionally, we also explore other alignment strategies referred to as "*All*" and "*Merge*". "*All*" indicates that we associate all pinyin characters with the correct Chinese character (e.g., "jin(今)" corresponding to "今今今"). "*Merge*" indicates that we merge the encoder representations of all pinyin characters that correspond to a Chinese character and forward the merged representation to the corrector. Table 8 shows the performance of PESC on the TAL test set using different alignment strategies. A comparison reveals that our method outperforms "*All*" and "*Merge*" in terms of performance. Particularly, at correction-level, our method achieves a superior F1 score of 1.69 over "*All*" and 1.03 over "*Merge*". We posit that both "*All*" and "*Merge*" introduce challenges to model learning, consequently diminishing model performance. Additionally, due to the occurrence of insertion errors, "*All*" leads to incorrect alignment between pinyin characters and Chinese characters (e.g., insertion error "jimn(今)" leads "m" corresponding to "今"), thereby elucidating the inferior performance compared to "*Merge*".

6. Conclusion

In this paper, we introduce the Realistic Chinese Spell Checking (RCSC) task, which aims to convert the realistic Chinese text into the corresponding correct text. Compared to existing tasks, RCSC has higher capability requirements for models that need to address both Chinese misspellings and pinyin conversions simultaneously. To benchmark the performance of various methods on RCSC, we present the RCSCB, which consists of 581,657 samples in total. To address the unique challenges of RCSC better, we also propose a PESC model with elaborately designed strategies. Experimental results show that PESC can achieve state-of-the-art performance on RCSCB. The ablation study results demonstrate the effectiveness of each strategy of the PESC. We hope more progress can be made in this more realistic CSC task, which has significant application value in real-world scenarios.

7. Acknowledgements

This work is supported by the National Science Foundation of China (No. 62206194), and the Natural Science Foundation of Jiangsu Province (No. BK20220488).

8. Bibliographical References

Tao-Hsing Chang, Hsueh-Chih Chen, and Cheng-Han Yang. 2015. Introduction to a proofreading tool for chinese spelling check task of sighan-8. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 50–55.

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to chinese pinyin input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 241–247.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 871–881.

Wei-Cheng Chu and Chuan-Jie Lin. 2015. Ntou chinese spelling check system in sighan-8 bake-off. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 137–143.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746*.

Zheng Fang, Ruiqing Zhang, Zhongjun He, Hua Wu, and Yanan Cao. 2022. Non-autoregressive chinese asr error correction with phonological training. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5907–5917.

Zhao Guo, Yuan Ni, Keqiang Wang, Wei Zhu, and Guotong Xie. 2021. Global attention decoder for chinese spelling error correction. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1419–1428.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple, powerful chinese spell checker based on dae-decoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169.

Yan Hu, Iqra Ameer, Xu Zuo, Xueqing Peng, Yujia Zhou, Zehan Li, Yiming Li, Jianfu Li, Xiaoqian Jiang, and Hua Xu. 2023. Zero-shot clinical entity recognition using chatgpt. *arXiv preprint arXiv:2303.16416*.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon ime: neural-based chinese pinyin aided input method with customizable association. In *Proceedings of ACL 2018, System Demonstrations*, pages 140–145.

Tuo Ji, Hang Yan, and Xipeng Qiu. 2021. Spellbert: A lightweight pretrained model for chinese spelling check. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3544–3551.

Zhongye Jia and Hai Zhao. 2014. A joint graph model for pinyin-to-chinese conversion with typo correction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1512–1523.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *arXiv preprint arXiv:2301.08745*.

Yichong Leng, Xu Tan, Linchen Zhu, Jin Xu, Renqian Luo, Linquan Liu, Tao Qin, Xiangyang Li, Edward Lin, and Tie-Yan Liu. 2021. Fastcorrect: Fast error correction with edit alignment for automatic speech recognition. *Advances in Neural Information Processing Systems*, 34:21708–21719.

Chong Li, Cenyuan Zhang, Xiaoqing Zheng, and Xuan-Jing Huang. 2021. Exploration and exploitation: Two ways to improve chinese spelling correction models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 441–446.

Yinghui Li, Qingyu Zhou, Yangning Li, Zhongli Li, Ruiyang Liu, Rongyi Sun, Zizhen Wang, Chao Li, Yunbo Cao, and Hai-Tao Zheng. 2022. The past mistake is the future wisdom: Error-driven contrastive probability optimization for chinese spell checking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3202–3213.

Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. 2023. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*.

Shulin Liu, Shengkang Song, Tianchi Yue, Tao Yang, Huihui Cai, TingHao Yu, and Shengli Sun. 2022. Craspell: A contextual typo robust approach to improve chinese spelling correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3008–3018.

Shulin Liu, Tao Yang, Tianchi Yue, Feng Zhang, and Di Wang. 2021. Plome: Pre-training with misspelled knowledge for chinese spelling correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2991–3000.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam.

OpenAI. 2023. Introducing chatgpt.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Hui Su, Weiwei Shi, Xiaoyu Shen, Zhou Xiao, Tuo Ji, Jiarui Fang, and Jie Zhou. 2022. Rocbert: Robust chinese bert with multimodal contrastive pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 921–931.

Minghuan Tan, Yong Dai, Duyu Tang, Zhangyin Feng, Guoping Huang, Jing Jiang, Jiwei Li, and Shuming Shi. 2022. Exploring and adapting Chinese GPT to Pinyin input method. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 1899–1909, Dublin, Ireland. Association for Computational Linguistics.

Baoxin Wang, Wanxiang Che, Dayong Wu, Shijin Wang, Guoping Hu, and Ting Liu. 2021. Dynamic connected networks for chinese spelling check. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2437–2446.

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018. A hybrid approach to automatic corpus generation for chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zhixu Li, Jianfeng Qu, and Jie Zhou. 2023. Cross-lingual summarization via chatgpt. *arXiv preprint arXiv:2302.14229*.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *arXiv preprint arXiv:2303.13648*.

Shaohua Wu, Xudong Zhao, Tong Yu, Rongguo Zhang, Chong Shen, Hongli Liu, Feng Li, Hong Zhu, Jiangang Luo, Liang Xu, et al. 2021. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning. *arXiv preprint arXiv:2110.04725*.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at sighan bake-off 2013. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 35–42.

Heng-Da Xu, Zhongli Li, Qingyu Zhou, Chao Li, Zizhen Wang, Yunbo Cao, He-Yan Huang, and Xian-Ling Mao. 2021. Read, listen, and see: Leveraging multimodal information helps chinese spell checking. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 716–728.

Junjie Yu and Zhenghua Li. 2014. Chinese spelling error detection and correction based on language model, pronunciation, and shape. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 220–223.

Liang-Chih Yu, Lung-Hao Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2014. Overview of sighan 2014 bake-off for chinese spelling check. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 126–132.

Yong Deng Yunjie Ji, Yiping Peng Yan Gong, Baochang Ma Qiang Niu, and Xiangang Li. 2023. Belle: Be everyone's large language model engine. https://github.com/LianjiaTech/BELLE.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Ruiqing Zhang, Chao Pang, Chuanqiang Zhang, Shuohuan Wang, Zhongjun He, Yu Sun, Hua Wu, and Haifeng Wang. 2021. Correcting chinese spelling errors with phonetic pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2250–2261.

Shaohua Zhang, Haoran Huang, Jicong Liu, and Hang Li. 2020. Spelling error correction with soft-masked bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 882–890.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2019. Open vocabulary learning for neural Chinese Pinyin IME. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1584–1594, Florence, Italy. Association for Computational Linguistics.