WSpeller: Robust Word Segmentation for Enhancing Chinese Spelling Check

Fangfang Li¹, Youran Shan¹, Junwen Duan¹,∗, Xingliang Mao², MinLie Huang³

¹School of Computer Science and Engineering, Central South University
²Institute of Big Data And Internet Innovation, Hunan University of Technology and Business
³Beijing National Research Center for Information Science and Technology, Tsinghua University
{lifangfang,shanyouran,jwduan}@csu.edu.cn
xingliangmao0929@163.com
aihuang@tsinghua.edu.cn

Abstract

Chinese spelling check (CSC) detects and corrects spelling errors in Chinese texts. Previous approaches have combined character-level phonetic and graphic information, ignoring the importance of segment-level information. According to our pilot study, spelling errors are always associated with incorrect word segmentation. When appropriate word boundaries are provided, CSC performance is greatly enhanced. Based on these findings, we present WSpeller, a CSC model that takes into account word segmentation. A fundamental component of WSpeller is a W-MLM, which is trained by predicting visually and phonetically similar words. Through modification of the embedding layer’s input, word segmentation information can be incorporated. Additionally, a robust module is trained to assist the W-MLM-based correction module by predicting the correct word segmentations from sentences containing spelling errors. We evaluate WSpeller on the widely used benchmark datasets SIGHAN13, SIGHAN14, and SIGHAN15. Our model is superior to state-of-the-art baselines on SIGHAN13 and SIGHAN15 and maintains equal performance on SIGHAN14.

1 Introduction

Chinese Spelling Check (CSC) aims to detect and correct spelling errors in Chinese text, and can be used in many other natural language processing (NLP) applications, including search optimization (Martins and Silva, 2004; Gao et al., 2010), location extraction (Middleton et al., 2018), optical character recognition (OCR) (Afli et al., 2016), Automatic Speech Recognition (ASR) (Chao and Chang, 2020) and text classification (Xu et al., 2021).

Typos in the text are mostly phonetic or graphic of the correct character (Liu et al., 2010). Previous work focus on how to use the features of pronunciation and shape (Cheng et al., 2020; Ji et al., 2021). However, little attention has been devoted to the incorrect word segmentation induced by typos. As shown in Table 1, several typos not only affect semantic but also affect word segmentation. In this case, we perform a preliminary experiment detailed in Sec 3 to determine the impact of word segmentation on CSC. Experimental results indicate that if the appropriate word boundaries are provided, CSC performance will improve dramatically. However, it is also challenging to achieve robust word segmentations in the presence of typos.

To cope with the problem, we propose WSpeller (Word Speller), which consists of two submodules, namely the word segmentation module (WS-Module) and the correction module (C-Module). The WS-Module predicts the right word boundaries based on the original text (may have typos). Following that, the segment-level information is provided to C-Module for improving correction. The two submodules adopt different text encoding modules, and the hidden state of the WS-Module is passed to the C-Module, where the two submodules interact and are trained in a multi-task manner for mutual benefit. W-MLM (Word-Masked Language Model) is proposed as the text encoding module of C-Module because it adapts BERT (Devlin et al., 2019) with different embedding information and special replacement strategies and pre-trained on a large Chinese Corpus. WSpeller
is evaluated against three widely used datasets: SIGHAN13 (Wu et al., 2013), SIGHAN14 (Yu et al., 2014), SIGHAN15 (Tseng et al., 2015). The experimental results show that WSpeller improves the performance of CSC. WSpeller increased the F1 scores by 3% and 2.1% in SIGHAN13 and SIGHAN15 and achieved almost the same performance as the state-of-the-art on SIGHAN14. Ablation studies reveal that both the word segmentation and the W-MLM contribute considerably to the performance. The contribution of this paper is summarized as follows:

- We are among the first to use the robust word segmentation in CSC, and our findings confirm its efficacy.
- We propose W-MLM to improve task adaptability, in which the embedding and replacement strategy of the masked language model is tailored to the specific needs of the CSC task.
- Experiments on the SIGHAN benchmark datasets demonstrate that our method outperformed strong competitors.

## 2 Related Work

In recent years, CSC has received widespread attention. Early works focus on rules and confusion sets to deal with CSC, follow pipelines, including detection, candidate generation and selection (Xie et al., 2015; Xin et al., 2014; Zhang et al., 2015; Chen et al., 2013). These methods are limited by manually set rules and fixed confusion sets.

With the development of deep learning, most methods focus on how to integrate the phonetic and graphic features of characters into the model. Hong et al. (Hong et al., 2019) proposed FASPell, which exploited the phonetic and graphic information to exclude candidates with low similarity. Cheng et al. (Cheng et al., 2020) proposed SpellGCN, which employed GCN to combine the phonetic and graphic information as the output layer. Ji et al. (Ji et al., 2021) proposed SpellBERT, which used GCN to model the Pinyin and radical of characters as the embedding layer. Wang et al. (Wang et al., 2021) proposed DCN, which added phonetic embedding when generating candidates, used attention mechanism to calculate the fraction between adjacent characters, and obtained the best path. Bao et al. (Bao et al., 2020) proposed to use semantic candidates to word embedding represents the word segmentation of the text.

Some methods focus on training skills and the positive impact of detection on correction. Zhang et al. (Zhang et al., 2020) adopted GRU as detection network and BERT as correction network, and propose a soft masking strategy. Gan et al. (Gan et al., 2021) propose to judge the difficulty of learning samples through loss and apply self-supervised curriculum learning to CSC.

In recent years, the methods focus on integrating the similarity between characters into the model. However, they ignore the importance of word segmentation to CSC. In this paper, the word segmentation information of text is integrated into the model through pre-training and fine-tuning. Only a simple model structure can achieve a good correction effect.

## 3 Preliminary Experiment

Our primary premise is that the presence of typos hurts word segmentation and the ability correct would improve if better word boundaries were available. To validate this, we conduct a preliminary experiment in which we inject the precise word segmentation into BERT. As shown in Figure 1, the boundaries of words are represented by 0s and 1s, where 1 indicates segmentation is required before the current character. The word segmentation is obtained by LAC.

![Figure 1: Change of the embedding layer. The sequence corresponding to word embedding represents the word segmentation of the text.](https://pypi.org/project/LAC/)
Table 2: Results of preliminary experiments. BERT means the conventional bert-based method. Seg means the method to change the input information of the embedding layer.

<table>
<thead>
<tr>
<th>Method</th>
<th>Detection Level</th>
<th>Correction Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre</td>
<td>Rec</td>
</tr>
<tr>
<td>BERT</td>
<td>75.8</td>
<td>74.8</td>
</tr>
<tr>
<td>Seg</td>
<td>79.9</td>
<td>79.9</td>
</tr>
</tbody>
</table>

We, therefore, propose WSpeller, which includes word segmentation information for CSC. However, obtaining correct word boundaries for sentences containing typos is a challenge. WSpeller has been further enhanced with a robust WS-Module so that it can predict the word boundaries correctly.

4 Problem Definition

CSC aims to detect and correct typos in a given text $X = (x_1, x_2, \cdots, x_n)$, and finally get the detection sequence $D = (d_1, d_2, \cdots, d_n)$ representing whether corresponding $x_i$ is a typo and the correction $Y = (y_1, y_2, \cdots, y_n)$ without spelling error.

5 The Proposed Model

In this section, we present the WSpeller, whose overall architecture is illustrated in Figure 2. WSpeller consists of a WS-Module (5.1) and a C-Module (5.3). While the WS-Module aims to predict the correct word boundaries in spite of spelling errors, the C-Module aims to correct the typos in conjunction with the WS-Module. To fully exploit the word segmentation information, the C-Module uses a pre-trained language model W-MLM (5.2).

5.1 Word Segmentation Module

In this section, WS-Module which aims to predict the word segmentations of corrected text from source text is introduced in detail. Given a source text $X$, we can easily get the hidden state $H$ by BERT. Then we can get the predicted word segmentation result $S = (s_1, s_2, \cdots, s_n)$ by word segmentation network. The method is as follows:

$$H = BERT(X)$$
$$S = \text{Softmax}(W_1H + b_1)$$

where $W_1$ and $b_1$ are learnable matrices. Each item $s_i \in \{0, 1\}$ in $S$ indicates the word boundaries, and $s_i = 1$ means that word segmentation is required before the current character $x_i$. In the training phase, we use LAC to segment the target text $Y$ as the label of the WS-Module.

5.2 Pre-training W-MLM

The text encoding module of the C-Module is W-MLM, whose overall structure is shown in Figure 3. The skeleton of W-MLM is BERT. Since the text is always single sentence input in CSC, the W-MLM ignores the next sentence prediction (NSP).

The key points of W-MLM lie in three parts, including replacement strategy selection, replacement character generation and word segmentation information integration. Replacement strategy selection and replacement character generation are used to generate training sets.

5.2.1 Replacement Strategy Selection

In order for W-MLM to learn the knowledge of continuous errors, misuse of correct words and similarity between characters, different from the conventional replacement, we introduce the re-
We are prone to mistype characters as similar and words and similar characters. Inspired by this, we comprehensively consider similarity and commonality to generate candidates. For a character $ch$ to be replaced, its initial candidate list is the vocabulary. Firstly, we obtain the phonetic and graphic similarity between each candidate and $ch$ by calculating the edit distance of ideographic description sequence (IDS) and Pinyin sequence (Hong et al., 2019). At the same time, the frequency of candidates is counted in SogouCA2.

Then, we initialize the phonetic candidate list $\text{Candidate}_{p} = (cp_{1}, cp_{2}, \cdots, cp_{k})$ and graphic candidate list $\text{Candidate}_{g}$ for $ch$ according to the rule of preferentially taking the characters with higher similarity and higher frequency. We empirically set $k$ to 30, and update the frequency list $\text{Count}_{p} = (\text{count}_{p_{1}}, \text{count}_{p_{2}}, \cdots, \text{count}_{p_{k}})$ and $\text{Count}_{g}$. After that, we comprehensively consider similarity and commonality to obtain the phonetic score $\text{Score}_{P} = (sp_{1}, sp_{2}, \cdots, sp_{m})$ and graphic score $\text{Score}_{G}$. $\text{Score}_{G}$ is calculated in the same way as $\text{Score}_{P}$. Take $sp_{1}$ as an example:

$$\text{score}^{e}_{i} = \frac{\text{count}^{p}_{i} - \min(\text{Count}^{p})}{\max(\text{Count}^{p}) - \min(\text{Count}^{p})}$$ (3)

$$\text{score}^{e}_{i} = \text{Pronunciation}(cp_{i}, ch)$$ (4)

$$sp_{i} = w_{c} \times \text{score}^{e}_{i} + w_{s} \times \text{score}^{e}_{i}$$ (5)

where $\max(\text{Count}^{p})$ and $\min(\text{Count}^{p})$ is the Max and Min in $\text{Count}^{p}$. $\text{score}^{e}_{i}$, $\text{score}^{e}_{i}$ is the the frequency and similarity score. Pronunciation is to calculate the pronunciation similarity between $cp_{i}$ and $ch$. $w_{c}$ and $w_{s}$ is the weights of $\text{score}^{e}_{i}$, $\text{sim}^{p}_{i}$. We empirically set them to 0.3 and 0.7.

Finally, the higher the score $sp_{i}$ of $cp_{i}$, the more likely $cp_{i}$ is to be used to replace $ch$ in phonetically similar character replacement, as are other replacement strategies.

2http://www.sogou.com/labs/resource/ca.php

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Prob</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin&amp;Same</td>
<td>10%</td>
<td>跟我一起(qi)去爬山吗</td>
</tr>
<tr>
<td>[MASK]</td>
<td>20%</td>
<td>跟我一[MASK]去爬山吗</td>
</tr>
<tr>
<td>Random</td>
<td>10%</td>
<td>跟我一里(li)去爬山吗</td>
</tr>
<tr>
<td>Graphics</td>
<td>30%</td>
<td>跟我一气(qi)去爬山吗</td>
</tr>
<tr>
<td>Pronunciation</td>
<td>10%</td>
<td>跟我一气(qi)去爬山吗</td>
</tr>
<tr>
<td>Continuous</td>
<td>10%</td>
<td>跟我一气(qi)去爬山吗</td>
</tr>
</tbody>
</table>

Table 3: Probability of each replacement strategy being selected. The replaced characters are marked in red, and the pronunciation of it is in parentheses. Origin&Same means the original text and the same character replacement. [MASK] means [MASK] replacement. Random means random character replacement. Graphics means graphically similar character replacement. Pronunciation means phonetically similar character replacement. Continuous means continuous similar character replacement and correct word replacement.
5.2.3 Word Segmentation Information Integration

In WS-Module, the text is input as a single sentence. Therefore, the segment sequence of the original BERT for the sentence number is actually all zero. Obviously, it does not contain any information useful for correction. In this regard, we integrate word segmentation information into W-MLM by replacing segmentation embedding with word embedding. The word segmentation results $S$ of the text before random replacement is obtained by LAC. Thus, the model can learn the changes of embedding layer in the pre-training stage, so that the model can perform better in the fine-tuning stage.

5.3 Correction Module

As shown in Figure 2, the C-Module is composed of W-MLM and correction network. Different from the WS-Module, we have obtained the predicted word segmentation result $S$ at this stage. We assume that $S$ is the word segmentation of the target text $Y$, and this information will play a great role in correction.

Hidden state $\tilde{H}$ can be obtained by token sequence $T$, word segmentation $S$, and position sequence $P$. In addition, we believe that C-Module should have a positive impact on WS-Module. Therefore, we add the $H$ of the WS-Module and the $\tilde{H}$ of the C-Module as the final hidden state $H'$. After that, the target text $Y$ is further obtained:

\[
\tilde{H} = W\text{-}MLM(T, S, P) \quad (6)
\]
\[
H' = H + \tilde{H} \quad (7)
\]
\[
\tilde{H}' = \text{LayerNorm}(\text{GELU}(W_2\tilde{H}' + b_2)) \quad (8)
\]
\[
Y = \text{Softmax}(W_3\tilde{H}' + b_3) \quad (9)
\]

where $W_2, W_3, b_2, b_3$ are learnable parameters, LayerNorm and GELU represent the layer normalization and activation functions in BERT, respectively.

5.4 Learning

WS-Module and C-Module are jointly trained. The learning process is driven by optimizing correction and word segmentation respectively.

\[
\mathcal{L}_s = - \sum_{i=1}^{n} \log P_s(s_i = \text{truth}_{si}|X) \quad (10)
\]
\[
\mathcal{L}_c = - \sum_{i=1}^{n} \log P_c(y_i = \text{truth}_{yi}|X) \quad (11)
\]
\[
\mathcal{L} = \lambda \times \mathcal{L}_s + (1 - \lambda) \times \mathcal{L}_c \quad (12)
\]

Table 4: Statistics of the datasets. ASL is the average length of sentences

<table>
<thead>
<tr>
<th></th>
<th>Sentences</th>
<th>ASL</th>
<th>Typos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid</td>
<td>271,329</td>
<td>42.6</td>
<td>381,962</td>
</tr>
<tr>
<td>SIGHAN13</td>
<td>700</td>
<td>41.8</td>
<td>343</td>
</tr>
<tr>
<td>SIGHAN14</td>
<td>3,437</td>
<td>49.6</td>
<td>5,122</td>
</tr>
<tr>
<td>SIGHAN15</td>
<td>2,338</td>
<td>31.3</td>
<td>3,037</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>277,804</strong></td>
<td><strong>42.6</strong></td>
<td><strong>390,464</strong></td>
</tr>
</tbody>
</table>

where $\mathcal{L}_s$ is the loss of WS-Module, $\text{truth}_{si}$ is the correct result of $s_i$. $\mathcal{L}_c$ is the loss of C-Module, $\text{truth}_{yi}$ is the correct result of $y_i$. $\lambda \in [0, 1]$ is the weight of $\mathcal{L}_s$, which represents the focus of model training. When $\lambda$ is closer to 1, it means that more attention is paid to the WS-Module.

6 Experiments

6.1 Dataset and Metrics

In the pre-training stage, we used many pre-processed data from Wikipedia used in TaCL (Su et al., 2021). In the fine-training, the data used for training includes all the data in Hybrid (Wang et al., 2018) and the training sets in SIGHAN13 (Wu et al., 2013), SIGHAN14 (Yu et al., 2014), and SIGHAN15 (Tseng et al., 2015). To evaluate, we use the test sets of SIGHAN13, SIGHAN14, and SIGHAN15. Since the SIGHAN is traditional, we use the processed data\(^3\) which follow the previous work (Cheng et al., 2020) and convert them to Simplified Chinese using the OpenCC tool\(^4\).

The statistics for the datasets are shown in Table 4.

We report Precision, Recall, and F1 scores at sentence level in detection and correction, which are commonly used in the CSC. Sentence level means that the current sentence can only be judged to be correct if all typos have been detected and corrected.

6.2 Baseline

We compare WS-Pre to five typical baselines.

---

\(^3\)https://github.com/DaDaMrX/RealiSe

\(^4\)https://github.com/BYVoid/OpenCC
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Detection Level Pre</th>
<th>Rec</th>
<th>F1</th>
<th>Correction Level Pre</th>
<th>Rec</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGHAN13</td>
<td>FASPell (Hong et al., 2019)</td>
<td>76.2</td>
<td>63.2</td>
<td>69.1</td>
<td>73.1</td>
<td>60.5</td>
<td>66.2</td>
</tr>
<tr>
<td></td>
<td>SpellGCN (Cheng et al., 2020)</td>
<td>80.1</td>
<td>74.4</td>
<td>77.2</td>
<td>78.3</td>
<td>72.7</td>
<td>75.4</td>
</tr>
<tr>
<td></td>
<td>SpellGCN* (Wang et al., 2021)</td>
<td>85.2</td>
<td>77.7</td>
<td>81.2</td>
<td>83.4</td>
<td>76.1</td>
<td>79.6</td>
</tr>
<tr>
<td></td>
<td>DCN* (Wang et al., 2021)</td>
<td><strong>86.8</strong></td>
<td>79.6</td>
<td>83.0</td>
<td>86.7</td>
<td>77.7</td>
<td>81.0</td>
</tr>
<tr>
<td></td>
<td>BERT*</td>
<td>81.7</td>
<td>86.0</td>
<td>83.8</td>
<td>79.7</td>
<td>83.9</td>
<td>81.8</td>
</tr>
<tr>
<td></td>
<td>WSpleller*</td>
<td>82.3</td>
<td><strong>86.9</strong></td>
<td><strong>84.6</strong></td>
<td>81.2</td>
<td><strong>85.7</strong></td>
<td><strong>83.4</strong></td>
</tr>
<tr>
<td>SIGHAN14</td>
<td>FASPell (Hong et al., 2019)</td>
<td>61.0</td>
<td>53.5</td>
<td>57.0</td>
<td>59.4</td>
<td>52.0</td>
<td>55.4</td>
</tr>
<tr>
<td></td>
<td>SpellGCN (Cheng et al., 2020)</td>
<td>65.1</td>
<td>69.5</td>
<td>67.2</td>
<td>63.1</td>
<td>67.2</td>
<td>65.3</td>
</tr>
<tr>
<td></td>
<td>DCN (Wang et al., 2021)</td>
<td>67.4</td>
<td><strong>70.4</strong></td>
<td><strong>68.9</strong></td>
<td>65.8</td>
<td><strong>68.7</strong></td>
<td><strong>67.2</strong></td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td>66.9</td>
<td>63.9</td>
<td>65.4</td>
<td>64.6</td>
<td>61.7</td>
<td>63.1</td>
</tr>
<tr>
<td></td>
<td>WSpleller</td>
<td><strong>70.4</strong></td>
<td>66.3</td>
<td>68.3</td>
<td><strong>69.0</strong></td>
<td>65.0</td>
<td>67.0</td>
</tr>
<tr>
<td>SIGHAN15</td>
<td>FASPell (Hong et al., 2019)</td>
<td>67.6</td>
<td>60.0</td>
<td>63.5</td>
<td>66.6</td>
<td>59.1</td>
<td>62.6</td>
</tr>
<tr>
<td></td>
<td>Soft-Masked BERT (Zhang et al., 2020)</td>
<td>73.7</td>
<td>73.2</td>
<td>73.5</td>
<td>66.7</td>
<td>66.2</td>
<td>66.4</td>
</tr>
<tr>
<td></td>
<td>SpellGCN (Cheng et al., 2020)</td>
<td>74.8</td>
<td>80.7</td>
<td>77.7</td>
<td>72.1</td>
<td>77.7</td>
<td>75.9</td>
</tr>
<tr>
<td></td>
<td>DCN (Wang et al., 2021)</td>
<td>77.1</td>
<td><strong>80.9</strong></td>
<td>79.0</td>
<td>74.5</td>
<td><strong>78.2</strong></td>
<td>76.3</td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td>78.7</td>
<td>74.5</td>
<td>76.5</td>
<td>75.8</td>
<td>71.7</td>
<td>73.7</td>
</tr>
<tr>
<td></td>
<td>WSpleller</td>
<td><strong>81.9</strong></td>
<td>78.0</td>
<td><strong>79.9</strong></td>
<td><strong>79.9</strong></td>
<td>76.1</td>
<td><strong>77.9</strong></td>
</tr>
</tbody>
</table>

Table 5: Results of Precision, Recall, and F1 scores(%). * indicates that “的”, “得”, “地” are ignored when calculating results on SIGHAN13, and the results of DCN* and SpellGCN* are reported by DCN (Wang et al., 2021).

- **FASPell (Hong et al., 2019)** measures the similarity between characters and filters characters with low similarity.
- **SpellGCN (Cheng et al., 2020)** uses GCN to model the similarity between characters and integrates the knowledge into the output layer.
- **Soft-Masked BERT (Zhang et al., 2020)** proposes a Soft-Mask layer that adjusts the embedding according to the probability of typos.
- **SpellBERT (Ji et al., 2021)** uses GCN to model the Pinyin and radical of characters as the embedding layer.
- **DCN (Wang et al., 2021)** adds pronunciation information when generating candidates, and models the correlation between adjacent characters to select the best candidates.

### 6.3 Experiment Setting

We use one Tesla V100 for pre-training and two GeForce RTX 2080Ti for follow-up experiments. In the fine-tuning stage, maximum sequence length is set to 160, batch size is set to 22, learning rate is set to 5e-5, training epoch is set to 20, warmup is set to 0.1, $\lambda$ is set to 0.2, optimization method is Adam. In pre-training stage, batch size is set to 32, learning rate is set to 1e-4, training epoch is set to 1, optimization method is Adam. In addition, there are lots of labeling errors about ”的”, “得”, “地” in SIGHAN13, which affect the evaluation, so we follow the previous work (Wang et al., 2021) to ignore all corrections about ”的”, “得”, “地”.

Furthermore, since the ability of W-Module is poor at the beginning of training, it will affect the training of C-Module. Therefore, we use scheduled sampling to provide a high proportion of correct word segmentation in the initial phase of training, and then slowly decrease the proportion until it reaches zero. In the first epoch, the probability of providing the correct word segmentation results is set to 0.9 and decreases to 0 in the 10th epoch.

### 6.4 Main Results

Table 5 illustrates the detection and correction performance on SIGHAN13, SIGHAN14, and SIGHAN15 of the proposed method and baseline models. As shown in Table 5, WSpleller significantly outperforms the other methods and the results show the effectiveness of our method and the enhancement of word segmentation for CSC.

Soft-Masked BERT uses a Soft-Masked layer
## 6.5 Ablation Study

Ablation study is presented to understand how the components of WSpeller and the hyperparameters influence the performance. The metrics reported in this section are averaged over SIGHAN13, SIGHAN14, and SIGHAN15.

### 6.5.1 Effect of Each Module

We remove the following components from WSpeller to study their contributions to the overall performance, and the results are shown in Table 7.

<table>
<thead>
<tr>
<th>Method</th>
<th>Detection Level</th>
<th>Correction Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>Pre</td>
</tr>
<tr>
<td>SpellGCN (Cheng et al., 2020)</td>
<td>83.7</td>
<td>85.9</td>
</tr>
<tr>
<td>SpellBERT (Ji et al., 2021)</td>
<td>-</td>
<td>87.5</td>
</tr>
<tr>
<td>DCN (Wang et al., 2021)</td>
<td>84.6</td>
<td>88.0</td>
</tr>
<tr>
<td>BERT</td>
<td>82.4</td>
<td>85.1</td>
</tr>
<tr>
<td>WSpeller</td>
<td>84.7</td>
<td>87.1</td>
</tr>
</tbody>
</table>

Table 6: Accuracy, Precision, Recall and F1 scores(%) at sentence level in detection and correction evaluated by SIGHAN15 official tools.

Further, we follow the previous works (Cheng et al., 2020; Wang et al., 2021) and use the official tool\(^5\) to evaluate the performance of WSpeller on SIGHAN15. The results are shown in Table 6. SpellBERT uses GCN to model the features of characters as the embedding layer. Compared with it, the F1 scores of WSpeller in detection and correction are increased by 3.9% and 4.1%. At the same time, WSpeller achieves the best result among all methods, which further indicates the effectiveness of WSpeller.

### 6.5.2 Effect of Each Module

We remove the following components from WSpeller to study their contributions to the overall performance, and the results are shown in Table 7.

<table>
<thead>
<tr>
<th>Method</th>
<th>Detection Level</th>
<th>Correction Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre</td>
<td>Rec</td>
</tr>
<tr>
<td>BERT</td>
<td>75.8</td>
<td>74.8</td>
</tr>
<tr>
<td>WSpeller</td>
<td>78.2</td>
<td>77.1</td>
</tr>
<tr>
<td>-W-MLM</td>
<td>76.4</td>
<td>75.9</td>
</tr>
<tr>
<td>-Different encoding</td>
<td>76.8</td>
<td>75.6</td>
</tr>
<tr>
<td>-Schedule sampling</td>
<td>77.2</td>
<td>76.5</td>
</tr>
<tr>
<td>LAC</td>
<td>77.9</td>
<td>77.1</td>
</tr>
</tbody>
</table>

Table 7: Ablation experiment results of Precision, Recall and F1 scores (%). BERT is BERT-based method. WSpeller is our proposed method. \(-W-MLM\) is to replace the W-MLM module with BERT. \(-Different\,\,encoding\) is to use the same text encoding module in WSpeller and C-Module. \(-Schedule\,\,sampling\) removes the training strategy. LAC integrates word segmentation information from the source text.

\(^5\)http://nlp.ee.ncu.edu.tw/resource/csc.html
the integrated word segmentation information contains some errors, its correction ability is worse than that of WSpeller, but higher than that of BERT. This indicates that the integration of word segmentation information is effective, and the accuracy of word segmentation information will affect the final correction effect, and WSpeller has stronger word segmentation ability in texts with typos.

### 6.5.2 Effect of Hyperparameter

In the joint training, \( \lambda \in [0, 1] \) represents the weight of WS-Module loss. When \( \lambda \) is large, the model will pay more attention to the training of WS-Module. Empirically, word segmentation is simpler than correction, so the weight of the WS-Module should be less than that of the C-Module in joint training.

We set \( \lambda \) to nine different values. When \( \lambda \) is set to a different value, the effect of WSpeller is shown in Figure 4. As shown in Figure 4, the Precision, Recall, and F1 score of the correction and detection of WSpeller reached the best when \( \lambda = 0.2 \). On the whole, when the loss of WS-Module accounts for small weight, WSpeller has better performance.

### 6.6 Case Study

Three representative examples of WSpeller are shown in Table 8. In the first example, the typo led to the wrong word segmentation. WSpeller not only correct the typo, but also get the correct word boundaries. In the second example, the typo led to the wrong word segmentation, but it is not labeled in the dataset. WSpeller correct the typo and get the correct word segmentation result. In the third example, “板” is correct, but is incorrectly labeled as “版”. In the last two examples, the prediction results of WSpeller are correct. However, due to mislabeling, these correct prediction would not improve the F1 scores, but reduce it. For further analysis, we manually count the proportion of text containing label errors according to the badcases of WSpeller.

In SIGHAN13, SIGHAN14, and SIGHAN15, 28.2%, 35.3%, and 30.6% of badcases contain label errors, which greatly affects the evaluation. The proportion of label errors in SIGHAN14 is the highest, which may be one of the reasons why the effect of WSpeller in SIGHAN14 is not satisfactory. Several examples show that WSpeller has better word segmentation ability when typos exist. Meanwhile, integrating word segmentation information contributes to the correction ability.

Further, according to our statistics, of the typos in three datasets, about 52.5%, 42.2%, and 39.5%, respectively, resulted in word segmentation errors. WSpeller can correct 84.3%, 66.6%, and 63.6% error word boundaries only based on source text respectively. This indicates that a considerable part of typos will affect word segmentation, and WSpeller has good word segmentation ability even
if there are typos in the text.

7 Conclusion

In this paper, we have proposed WSpeller, a novel end-to-end framework for CSC. Unlike most of the previous methods, which included the phonetic and graphic information of characters in the model through various methods, we emphasize the importance of robust word segmentation in the CSC task. We added an additional WS-Module to predict the correct word boundaries for sentences with typos. The experimental results also suggest that adding word segmentation information to the model can improve the performance.

In the future, we will investigate other methods for providing word segmentation information to the model for correction. We will also explore how to use WSpeller for Chinese semantic corrections.

8 Limitations

In our study, we discovered that correct word segmentation would improve the ability of correction, but achieving accurate word segmentation from source texts that contain typos remains a challenge. However, although WSpeller is capable of segmenting words with high accuracy, there is still a lot of room for improvement. By having more accurate word boundaries available, the model’s correction capabilities can be further enhanced, as the upper bound in the preliminary experimental results in Section 3. As WSpeller encodes text twice, its inference speed is limited. A future attempt will be made to improve WS-Module to reduce its complexity.

Acknowledgements

This research is supported by National Natural Science Foundation of China [62172449, 62006251], Hunan Provincial Natural Science Foundation of China [2021JJ30870,2022JJ30211], Changsha Municipal Natural Science Foundation [kj2202300]. And this research was carried out in part using computing resources at the High Performance Computing Center of Central South University.

References


Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple,


