PHMOSpell: Phonological and Morphological Knowledge Guided Chinese Spelling Check

Li Huang[1,2], Junjie Li[2], Weiwei Jiang[2], Zhiyu Zhang[2], Minchuan Chen[2], Shaojun Wang[2] and Jing Xiao[2]

[1]Fudan University
[2]Ping An Technology
lhuang9703@gmail.com, junjielee815@gmail.com

Abstract

Chinese Spelling Check (CSC) is a challenging task due to the complex characteristics of Chinese characters. Statistics reveal that most Chinese spelling errors belong to phonological or visual errors. However, previous methods rarely utilize phonological and morphological knowledge of Chinese characters or heavily rely on external resources to model their similarities. To address the above issues, we propose a novel end-to-end trainable model called PHMOSpell, which promotes the performance of CSC with multi-modal information. Specifically, we derive pinyin and glyph representations for Chinese characters from audio and visual modalities respectively, which are integrated into a pre-trained language model by a well-designed adaptive gating mechanism. To verify its effectiveness, we conduct comprehensive experiments and ablation tests. Experimental results on three shared benchmarks demonstrate that our model consistently outperforms previous state-of-the-art models.

1 Introduction

Chinese Spelling Check (CSC) is a fundamental task in Chinese Natural Language Processing (NLP), which aims to automatically detect and correct spelling errors in Chinese sentences. These errors typically consist of human writing errors and machine recognition errors by automatic speech recognition (ASR) or optical character recognition (OCR) systems (Yu et al., 2014). CSC serves as a preliminary component for other downstream tasks like information retrieval (IR) in search engine, thus significantly affects the final performance of these tasks.

Chinese is an ideograph language which contains numerous characters and has no between-word delimiters. These characteristics make its spelling check more difficult than other alphabetical languages such as English. Specifically, for error detection, Chinese words usually consist of several characters and have no clear word boundaries, which makes it impossible to detect spelling errors just using individual word or character. They must be put in a specific sentence to capture contextual semantic information. For error correction, how to select correct candidates from tremendous character sets remains a great challenge. In contrast to English words that are composed of a small set of alphabet letters, there are more than 10k Chinese characters, and 3.5k of them are frequently used (Wang et al., 2019b). Besides, unlike English, almost all Chinese spelling errors are real-word errors which means the misspelling one is also a valid character in the vocabulary. (Kukich, 1992; Jia et al., 2013; Yu and Li, 2014).

p-s error:	
wrong sentence:	人们必(pinyin: bi4)生去追求的目标。
ground truth:	人们毕(pinyin: bi4)生去追求的目标。
v-s error:	
wrong sentence:	迎接每一个固(radicals: 古,口)难。
ground truth:	迎接每一个困(radicals: 木,口)难。

Table 1: Examples of p-s (phonological similarity) error and v-s (visual similarity) error from SIGHAN13 (Wu et al., 2013). Here, the ground truth of the p-s error means "The goal that people pursue throughout their lives" and the ground truth of the v-s error means "Get prepared for every difficulty".

Since a great number of Chinese characters are similar either in phonology or morphology, they are easily misused with each other. According to (Liu et al., 2011), 76% of Chinese spelling errors belong to phonological similarity error and 46% belong to visual similarity error. Table 1 presents examples of these two common errors. The pronunciation and the shape of Chinese characters can be characterized by pinyin[1] and radicals[2], respectively.

[1]pinyin is the official phonetic system of Mandarin Chinese, which usually consists of three parts: initials, finals and tones.

[2]radical is the basic building blocks of all Chinese charac-

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5958–5967
August 1–6, 2021.

Previous methods have made attempts to fuse these two information into the process of CSC (Jin et al., 2014; Han et al., 2019; Hong et al., 2019; Nguyen et al., 2020). However, pinyin or radicals in these methods were used as external resources or heuristic filters and can not be trained with the model in an end-to-end style. More recently, Cheng et al. (2020) proposed SpellGCN, which incorporated phonological and morphological similarities into a pre-trained language model by graph convolutional network (GCN). However, their similarity graphs relied on specific confusion sets. Since confusion sets are unable to cover all characters, SpellGCN can only fuse limited information. Furthermore, they just used a simple aggregate strategy for feature fusion.

To tackle the above issues, we propose a novel framework called PHMOSpell. PHMOSpell incorporates pinyin and glyph features into a pre-trained language model via an adaptive gating module for CSC. These features are derived from intermediate representations of dominant Tacotron2 (Shen et al., 2018) in text-to-speech (TTS) task and VGG19 (Simonyan and Zisserman, 2014) in computer vision (CV) task. We combine them with semantic representation from a pre-trained language model by the proposed adaptive gating module, enabling the model to be trained end-to-end. Comprehensive experiments are conducted on three shared benchmarks to prove that latent representations in our method can capture not only semantic but also phonological and morphological information. Experimental results demonstrate that our method outperforms all baseline methods on three benchmarks.

The contributions of this paper are in three folds: 1) We derive both phonological and morphological knowledge of Chinese characters from multi-modality and apply them to CSC. 2) We design a novel adaptive gating mechanism, which effectively incorporates the multi-modal information into a pre-trained language model in an end-to-end trainable way. 3) We achieve state-of-the-art performance on three benchmark datasets using the proposed model.

2 Related Work

CSC has received active research in recent years. Previous studies on CSC can be divided into three categories: rule based methods, statistical based methods and deep learning based methods. Mangu and Brill (1997) proposed a rule based approach for automatically acquiring linguistic knowledge from a small set of easily understood rules. Jiang et al. (2012) arranged a new grammar system of rules to solve both Chinese grammar errors and spelling errors. Xiong et al. (2015)'s HANSpeller was based on an extended HMM, ranker based models and a rule based model. For statistical based methods, Noisy Channel Model (Brill and Moore, 2000, 2008; Chiu et al., 2014; Noaman et al., 2016; Bao et al., 2020) is the most widely used model. Statistical based methods usually narrowed the candidates choice by utilizing a predefined confusion set (Chen et al., 2013; Hsieh et al., 2013; Wang et al., 2019a), which contains a set of similar character pairs. These similar characters were used to replace each other and language models were leveraged to measure the quality of the modified sentences (Liu et al., 2013; Yu and Li, 2014; Xie et al., 2015). More recently, deep learning has achieved excellent results on many NLP tasks, including CSC. Wang et al. (2019a) proposed an end-to-end confusionset-guided encoder-decoder model, which treated CSC as a sequence-to-sequence task and infused confusion sets information by copy mechanism. FASpell (Hong et al., 2019) employed BERT (Devlin et al., 2019) as a denoising autoencoder (DAE) for CSC. SpellGCN (Cheng et al., 2020) constructed two similarity graphs over the characters in confusion sets and employed graph convolutional network on these two graphs to capture the pronunciation/shape similarities between characters. Soft-Masked BERT (Zhang et al., 2020) was proposed to combine a Bi-GRU based detection network and a BERT based correction network, where the former passed its prediction results to the latter using soft masking mechanism. Nguyen et al. (2020) applied TreeLSTM (Tai et al., 2015; Zhu et al., 2015) on the tree structure of the character radicals to get hierarchical character embeddings, which was used as an adaptable filtering component for candidates selection.

3 Approach

3.1 Problem Formulation

Generally, CSC can be regarded as a revision task on Chinese sentences. Given a Chinese sentence $X = \{x_1, x_2, ..., x_n\}$ of length n, the model needs to detect spelling errors on character level and output its correct corresponding sentence Y =

ters, there are about 216 different radicals in Chinese.

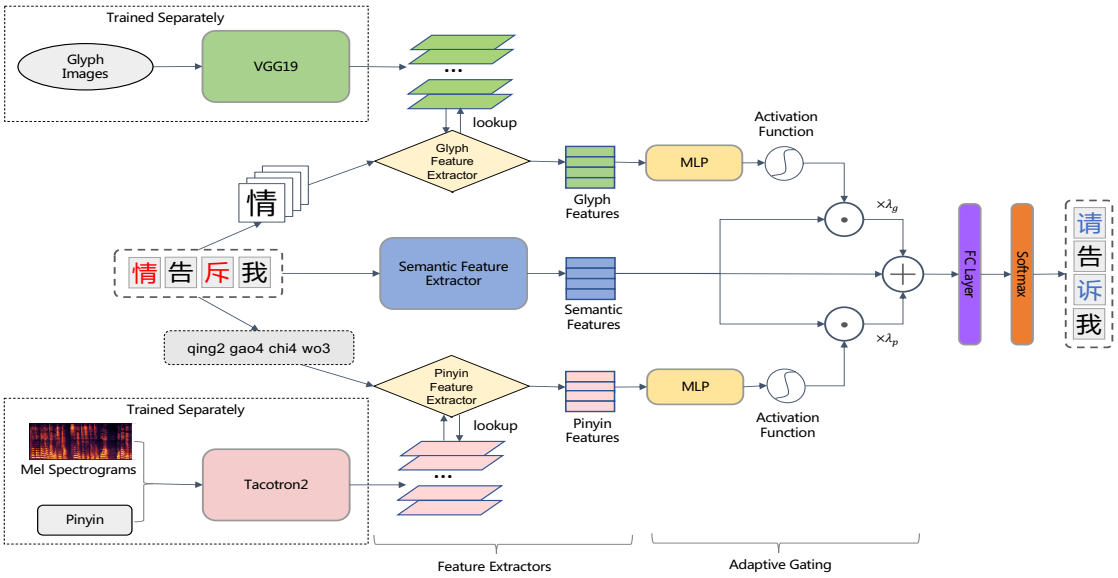

Figure 1: The architecture of our model. $\odot$ and $\oplus$ denote element-wise multiplication and addition operation, respectively. Correct sentence means 'Please tell me'.

$\{y_1, y_2, ..., y_n\}$. Although CSC can be viewed as a kind of sequence-to-sequence (Seq2Seq) task, it is different from other Seq2Seq tasks (e.g., Text Summarization, Machine Translation): the input and output sequences of the former are equal in length. Most or even all of the characters in the input sequence remain unchanged, only a few of them need to be corrected.

3.2 Model

Our model consists of three feature extractor modules and an adaptive gating module used to fuse kinds of features. Figure 1 illustrates the architecture of our model. Given a sentence, our model firstly extracts pinyin feature, glyph feature and context-sensitive semantic feature for every character, then integrates three features by the adaptive gating module. Finally, the integrated representation of each character is fed into a fully-connected layer to calculate the probabilities over the whole vocabulary, where the character with the highest probability is picked as the substitute.

In the following subsections, we will elaborate the implementation of each module.

3.3 Pinyin Feature Extractor

Neural TTS models, like Tacotron2 (Shen et al., 2018), have achieved high-quality performance in producing natural-sounding synthetic speech. We propose to generate the phonological representations of Chinese characters through a TTS model so that CSC can benefit from realistic pronunciation similarities between characters. In this paper, we leverage Tacotron2, a recurrent sequence-to-sequence mel spectrograms prediction network, to help modeling the phonological representations since its location-sensitive attention can create effective time alignment between the character sequence and the acoustic sequence. When training a Chinese TTS system with Tacotron2, characters are first converted to pinyin sequence as phoneme form. Then the sequence is represented by the encoder using an embedding layer and the hidden representations are consumed by the decoder to predict a corresponding mel spectrogram one frame at a time. Motivated by this, we train Tacotron2 separately using public Chinese female voice datasets [3] with teacher forcing. During training, we utilize pinyin transcription and mel spectrograms as input to help modeling pinyin representations. Then we extract pinyin embedding layer of the encoder as our pinyin feature extractor to generate the phonological representations for CSC. When given a Chinese sentence X, our model first converts it to a pinyin sequence using pypinyin[4]. Then dense feature for pinyin sequence $\mathbf{F}_p = \{\mathbf{f}_1^p, \mathbf{f}_2^p, ..., \mathbf{f}_n^p\}$ can be obtained by using pinyin feature extractor as a lookup table, where $\mathbf{f}_i^p \in \mathbb{R}^{d_p}$ and d_p is the dimen-

[3] https://test.data-baker.com/#/data/index/source

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

sion of the pinyin feature.

3.4 Glyph Feature Extractor

As Chinese characters are composed of graphical components, it is intuitive that the representations for Chinese characters could benefit from the spatial layout of these components. Motivated by Meng et al. (2019) and Sehanobish and Song (2019)'s exploration on using glyph images for Chinese named entity recognition (NER) and Chinese word segmentation (CWS), we employ a glyph feature extractor to extract glyph features for Chinese characters. We make use of 8106 Chinese glyph images released by (Sehanobish and Song, 2019). To take advantage of powerful pre-trained models and avoid training from scratch, VGG19 (Simonyan and Zisserman, 2014) pretrained on ImageNet is adopted as the backbone of the glyph feature extractor. Following (Meng et al., 2019), we further finetune it with the objective of recovering the identifiers from glyph images to solve the problem of domain adaptation. After that, we drop the last classification layer and use the outputs of VGG19's last max pooling layer as glyph features. For a given sentence X, our glyph feature extractor is able to first retrieve images for its characters and then generate glyph features: $\mathbf{F}^g = \{\mathbf{f}_1^g, \mathbf{f}_2^g, ..., \mathbf{f}_n^g\}$, where $\mathbf{f}_i^g \in \mathbb{R}^{d_g}$ is the glyph feature of the i_{th} character x_i and d_g is the dimension of the glyph feature.

3.5 Semantic Feature Extractor

Beyond the phonological and the morphological information, we adopt empirically dominant pre-trained language model to capture semantic information from context. Following (Hong et al., 2019; Cheng et al., 2020; Zhang et al., 2020), BERT is employed as the backbone of our semantic feature extractor. Given an input sentence X, the extractor outputs hidden states $\mathbf{F}^s = \{\mathbf{f}_1^s, \mathbf{f}_2^s, ..., \mathbf{f}_n^s\}$ at the final layer of BERT as semantic features, where $\mathbf{f}_i^s \in \mathbb{R}^{d_s}$ and d_s is the dimension of the semantic feature.

3.6 Adaptive Gating

Most previous methods for CSC simply used addition or concatenation to fuse different features. However, these fusion strategies ignore the relationship between the features. To tackle this issue, we propose an innovative adaptive gating mechanism served like a gate to finely control the fusion of features. It is defined as follows:

$$AG(\mathbf{F}^p, \mathbf{F}^s) = \sigma(\mathbf{F}^p\mathbf{W}^p + \mathbf{b}^p) \cdot \mathbf{F}^s \quad (1)$$

$$AG(\mathbf{F}^g, \mathbf{F}^s) = \sigma(\mathbf{F}^g\mathbf{W}^g + \mathbf{b}^g) \cdot \mathbf{F}^s \quad (2)$$

where $\mathbf{W}^p \in \mathbb{R}^{d_p \times d_s}, \mathbf{b}^p \in \mathbb{R}^{n \times d_s}, \mathbf{W}^g \in \mathbb{R}^{d_g \times d_s}, \mathbf{b}^g \in \mathbb{R}^{n \times d_s}$ are parameters to be learned. σ is a nonlinear activation function, which is a ReLU function in our implementation. "$\cdot$" represents element-wise multiplication. We employ the proposed gating mechanism to control how much information in pinyin and glyph features is fused with semantic feature and transferred to the next classifier module. The enriched feature $\mathbf{F}^e \in \mathbb{R}^{n \times d_s}$ is calculated as follows:

$$\mathbf{F}^e = \lambda_p \cdot AG(\mathbf{F}^p, \mathbf{F}^s) + \lambda_g \cdot AG(\mathbf{F}^g, \mathbf{F}^s) \quad (3)$$

where $\lambda_p + \lambda_g = 1$ are coefficients. Finally, we add residual connection to $\mathbf{F}^e$ and $\mathbf{F}^s$ by linear combination:

$$\mathbf{F}^{es} = \mathbf{F}^e + \mathbf{F}^s \quad (4)$$

3.7 Training

During the training process, the representation $\mathbf{F}^{es}$ is fed into a fully-connected layer for the final classification, which is defined as follows:

$$P(Y_p|X) = softmax(\mathbf{F}^{es}\mathbf{W}^{fc} + \mathbf{b}^{fc}) \quad (5)$$

where $\mathbf{W}^{fc} \in \mathbb{R}^{d_s \times V}, \mathbf{b}^{fc} \in \mathbb{R}^{n \times V}$ are learnable parameters for the fully-connected layer, V is the size of the vocabulary and Y_p is the predicted sentence given the erroneous sentence X.

The goal of training the model is to match the predicted sequence Y_p and the ground truth sequence Y_g. Overall, the learning process is driven by minimizing negative log-likelihood of the characters:

$$\mathcal{L} = -\sum_{i=1}^{n} logP(\hat{y}_i = y_i|X) \quad (6)$$

where $\hat{y}_i, y_i$ are the i_{th} characters of Y_p and Y_g, respectively.

3.8 Inference

At inference time, we select candidates with the highest probability given by the model for each character's correction. As for detection task, it is accomplished by checking whether the picked candidate is different with the input character.

Training Data	# erroneous sent / sent	Avg.length
SIGHAN13	340 / 700	41.8
SIGHAN14	3358 / 3437	49.6
SIGHAN15	2273 / 2339	31.3
(Wang et al., 2018)	271009 / 271329	42.5
Total	276980 / 277805	42.5
Test Data	**# erroneous sent / sent**	**Avg.length**
SIGHAN13	971 / 1000	74.3
SIGHAN14	520 / 1062	50.0
SIGHAN15	541 / 1100	30.6

Table 2: Statistics of datasets.

4 Experiments

4.1 Datasets

To investigate the effectiveness of our proposed method, we conduct extensive experiments on three shared benchmark datasets for CSC task. Specifically, we make use of training datasets from SIGHAN13 (Wu et al., 2013), SIGHAN14 (Yu et al., 2014) and SIGHAN15 (Tseng et al., 2015). We also include 271K training samples automatically generated by OCR-based and ASR-based methods (Wang et al., 2018) as in (Cheng et al., 2020; Nguyen et al., 2020). We employ test datasets of SIGHAN13, SIGHAN14, SIGHAN15 for evaluation. Following the same data preprocessing procedure with (Cheng et al., 2020; Nguyen et al., 2020), characters in all SIGHAN datasets are converted to simplified form using OpenCC[5]. We adopt SIGHAN's standard split of training and test data. The detailed statistic of the data is presented in Table 2.

4.2 Baseline Methods

We compare our method against several advanced methods proposed recently to investigate the potential of our framework. They are listed below:

- FASPell (Hong et al., 2019): This method employs BERT as a denoising autoencoder to generate candidates for wrong characters and filters the visually/phonologically irrelevant candidates by a confidence-similarity decoder.

- SpellGCN (Cheng et al., 2020): This method learns the pronunciation/shape relationship between the characters by applying graph convolutional network on two similarity graphs. It predicts candidates for corrections by combining graph representations with semantic representations from BERT.

- HeadFilt (Nguyen et al., 2020): This method uses adaptable filter learned from hierarchical character embeddings to estimate the similarity between characters and filter candidates produced by BERT.

- BERT: This method finetunes BERT with the training data and selects the character with the highest probability for correction.

4.3 Evaluation Metrics

We adopt sentence-level metrics for evaluation, which are widely used in previous methods for CSC task. Sentence-level metrics are stricter than character-level metrics since all errors in a sentence need to be detected and corrected. Metrics including accuracy, precision, recall and F1 score are calculated for errors detection and correction, respectively.

4.4 Experimental Setup

Our model is implemented based on huggingface's pytorch implementation of transformers[6]. We initialize weights of the semantic feature extractor using bert-base-chinese and weights of the glyph feature extractor using pretrained VGG19 from torchvision library[7]. Weights of the adaptive gating are randomly initialized. We train our model using AdamW optimizer for 5 epochs with learning rate $1e^{-4}$. Batch size is 64 for training and 32 for evaluation. Best λ_p, λ_g are 0.6, 0.4 for SIGHAN13, 0.8, 0.2 for SIGHAN14 and SIGHAN15. We train Tacotron2 using its open-source implementation[8] for 130k steps with default parameters, except the decay step is set to 15000. The number of our pinyin is 1920 and the dimension of the pinyin feature is 512. Characters are written using Hei Ti font[9] in 8106 glyph images. We finetune VGG19 on glyph images for 50 epochs with a batch size 32 and a learning rate $5e^{-4}$. The dimension of the glyph feature is 25088. All experiments are conducted on 2 Tesla V100 with 16G memory.

4.5 Main Results

Table 3 presents the results of all methods on three test datasets. Our method outperforms all previous

[5] https://github.com/BYVoid/OpenCC

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

[7] https://github.com/pytorch/vision

[8] https://github.com/Rayhane-mamah/Tacotron-2

[9] Hei Ti font is a very formal sans serif font for Chinese writing.

Test dataset	Method	Detection Level				Correction Level			
		Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
SIGHAN13	FASpell (2019)	-	76.2	63.2	69.1	-	73.1	60.5	66.2
	SpellGCN (2020)	-	80.1	74.4	77.2	-	78.3	72.7	75.4
	HeadFilt (2020)	74.9	**100.0**	74.9	85.7	74.1	**100.0**	74.1	85.1
	BERT	70.6	98.7	70.6	82.3	67.8	98.6	67.8	80.4
	PHMOSpell	**77.1**	99.5	**76.8**	**86.7**	**75.4**	99.5	**75.1**	**85.6**
SIGHAN14	FASpell (2019)	-	61.0	53.5	57.0	-	59.4	52.0	55.4
	SpellGCN (2020)	-	65.1	**69.5**	67.2	-	63.1	**67.2**	65.3
	HeadFilt (2020)	74.2	82.5	61.6	70.5	73.5	82.1	60.2	69.4
	BERT	72.7	78.6	60.7	68.5	71.2	77.8	57.6	66.2
	PHMOSpell	**78.5**	**85.3**	67.6	**75.5**	**76.9**	**84.7**	64.3	**73.1**
SIGHAN15	FASpell (2019)	-	67.6	60.0	63.5	-	66.6	59.1	62.6
	SpellGCN (2020)	-	74.8	**80.7**	77.7	-	72.1	**77.7**	75.9
	HeadFilt (2020)	79.3	84.5	71.8	77.6	78.5	84.2	70.2	76.5
	BERT	79.9	84.1	72.9	78.1	77.5	83.1	68.0	74.8
	PHMOSpell	**82.6**	**90.1**	72.7	**80.5**	**80.9**	**89.6**	69.2	**78.1**

Table 3: Performances of our method and baseline methods, where accuracy (Acc.), precision (Prec.), recall (Rec.), F1 on detection level and correction level are reported (%). Best results are in **bold**.

methods and achieves new state-of-the-art performance on all three datasets. Compared with the best baseline method (HeadFilt), the improvements of our method are 1.0%, 5.0%, 2.9% on detection-level F1 and 0.5%, 3.7%, 1.6% on correction-level F1 respectively, which verifies the effectiveness of our method.

We observe that our method substantially outperforms SpellGCN on the precision and F1 scores, which indicates that our method is superior to SpellGCN in fusing similarity knowledge. Although SpellGCN incorporates such knowledge, it relies on a predefined confusion set, which limits its generalization. Firstly, similarity knowledge cannot be obtained adequately since the confusion set is limited and unable to cover all characters. Secondly, the confusion set is manually constructed and has no golden-standard, which may bring about cascading errors. Our method achieves better F1 scores than HeadFilt, apparently because HeadFilt only leverages morphological knowledge in its post-filtering component. Finally, our method consistently beats vanilla BERT on all three datasets in terms of all metrics, which demonstrates the importance of incorporating the phonological and morphological knowledge into the semantic space for the CSC task.

4.6 Ablation Study

To study the effectiveness of each component in our method, we carry out ablation tests on three datasets. All ablation experiments with pinyin and glyph features are conducted using equal weights for pinyin feature and glyph feature ($\lambda_p = \lambda_g$) to avoid unnecessary biases they bring. Table 4 presents the results. First, replacing adaptive gating with a simple aggregate strategy leads to worse performance for both detection and correction, which demonstrates the benefit of using adaptive gating. We then remove pinyin feature extractor or glyph feature extractor from the model. The performance degrades more when removing pinyin feature compared with removing glyph feature, which implies that phonological information is more crucial for CSC. This is consistent with the finding that most Chinese spelling errors are caused by phonological similarity (Liu et al., 2011). The result further degrades when removing both features and adaptive gating module, and this trend intuitively indicates that both phonological and morphological information contribute to the final performance.

4.7 Effect of Hyper Parameters

In this subsection, we conduct experiments to analyze the effect of weights of features and the dimension of the pinyin feature.

Figure 2 shows how different weights influence the performance of the model. In this comparison, the value of λ_p (λ_g) changes from 0.0 (1.0) to 1.0 (0.0) with the gap of 0.2. We plot the detection-level and correction-level F1 scores on three datasets in Figure 2. The results consistently show that our model performs better when λ_p is set larger (e.g., 0.6 for SIGHAN13, 0.8 for

Test dataset	Method	Detection Level				Correction Level			
		Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
SIGHAN13	PHMOSpell (w/o PGA)	70.6	98.7	70.6	82.3	67.8	98.6	67.8	80.4
	PHMOSpell (w/o GE)	76.1	99.1	76.1	86.1	74.9	99.0	**74.8**	85.2
	PHMOSpell (w/o PE)	71.9	98.9	71.8	83.2	69.5	98.8	69.3	81.5
	PHMOSpell (w/ AS)	71.6	99.4	71.1	82.9	70.3	99.4	69.8	82.0
	PHMOSpell	**77.2**	**99.5**	**76.9**	**86.8**	**75.1**	**99.5**	74.7	**85.4**
SIGHAN14	PHMOSpell (w/o PGA)	72.7	78.6	60.7	68.5	71.2	77.8	57.6	66.2
	PHMOSpell (w/o GE)	76.4	**83.6**	64.3	72.7	**75.3**	**83.1**	62.0	71.1
	PHMOSpell (w/o PE)	76.2	82.9	64.7	72.7	74.8	82.2	61.8	70.6
	PHMOSpell (w/ AS)	73.4	81.3	59.1	68.5	72.4	80.8	57.0	66.8
	PHMOSpell	**76.6**	82.4	**66.3**	**73.5**	**75.3**	81.8	**63.6**	**71.6**
SIGHAN15	PHMOSpell (w/o PGA)	79.9	84.1	**72.9**	78.1	77.5	83.1	68.0	74.8
	PHMOSpell (w/o GE)	81.2	**88.7**	70.7	78.7	**80.0**	**88.4**	68.2	77.0
	PHMOSpell (w/o PE)	81.0	88.3	70.7	78.5	79.5	87.9	67.7	76.5
	PHMOSpell (w/ AS)	78.9	87.5	66.5	75.6	77.9	87.2	64.7	74.3
	PHMOSpell	**81.3**	88.6	71.2	**79.0**	**80.0**	88.2	**68.4**	**77.1**

Table 4: Ablation results on three datasets. PHMOSpell (w/ AS) denotes replacing adaptive gating module with aggregate strategy for feature fusion. PHMOSpell (w/o PE) denotes model without pinyin feature extractor. PHMOSpell (w/o GE) denotes model without glyph feature extractor. PHMOSpell (w/o PGA) denotes model without pinyin, glyph feature extractor and adaptive gating, which is a vanilla BERT implementation.

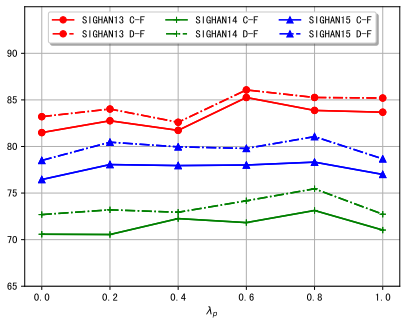

Figure 2: Effect of different weights for features. We show results (%) of detection-level F1 (D-F) and correction-level F1 (C-F) on three datasets.

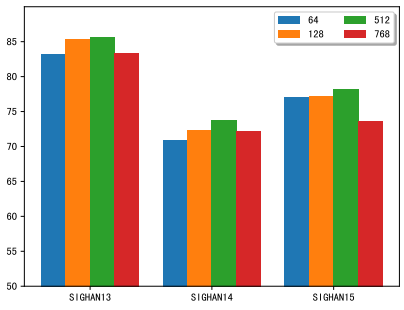

Figure 3: The results of correction-level F1 score (%) w.r.t. the dimension of the pinyin feature.

SIGHAN14, SIGHAN15), which means a higher weight on pinyin feature. Moreover, all of them outperform the model without any features.

Previous ablation tests show that the pinyin feature has more influence on the performance than the glyph feature. We further perform experiments by varying the dimension of the pinyin feature since it directly impacts the quality of the feature. Figure 3 shows larger dimensions perform better. However, it should be noted that the performance degrades when the dimension is larger than 512. This is reasonable due to the bias-variance phenomenon explained in (Yin and Shen, 2018). Feature with a small dimensionality can not capture all possible pinyin relations (high bias). On the other hand, feature with a large dimensionality includes too much noise (high variance). One must make a trade-off in dimensionality selection for high-quality features.

4.8 Features Visualization

To understand the effectiveness of our features more intuitively, we reduce features from high-dimensional space to low-dimensional space and visualize some of them using t-SNE (Van der Maaten and Hinton, 2008).

Figure 4 illustrates the embeddings of pinyin whose initial begins with “d”, “f”, “h” and “j”. One can find from the figure that embeddings form several clusters based on their pronunciations. Pinyin embeddings with more similar pronunciations (eg. “fu4” and “hu2”) are closer in distance than dissim-

ilar ones (eg. “hu2” and “dao4”). This suggests that the model has learned alignment between the pinyin feature and the realistic acoustic feature. We also plot glyph embeddings of characters with radical “口”, “土” at left side and characters with radical “口” at outside in Figure 5. They show the same trends as that of pinyin embeddings. Above all, this further verifies the effectiveness of both phonological and morphological knowledge derived from multi-modality.

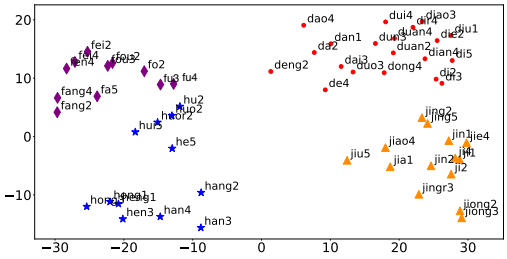

Figure 4: The scatter of similar pinyin in terms of pronunciation. Pinyin whose initial begins with “d”, “f”, “h”, “j” are shown in red, purple, blue, orange respectively.

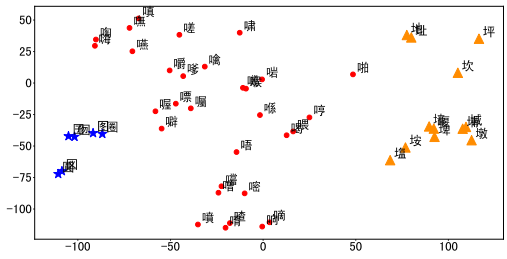

Figure 5: The scatter of similar characters in terms of shape. Characters with “口” and “土” at left side are shown in red and orange, characters with “口” at outside are shown in blue.

4.9 Discussion

To demonstrate how our model can handle phonological and visual errors, we showcase some representative cases from the test datasets. For instance, for the erroneous sentence “...不惜娱(pinyin: yu2)弄大臣...”, vanilla BERT corrects “娱弄” as “玩(pinyin: wan2) 弄 (play)” without considering phonological information, which is only semantically reasonable. Our model, however, takes both semantic and phonological knowledge into consideration and successfully generates a more proper correction “...不惜愚(pinyin: yu2) 弄大臣... (...Not hesitate to fool the minister...)”. Another case is “...那别人的欢(radicals: 又,欠) 说是没办法改变你的...”. Our model is capable of modifying it into correct sentence “...那别人的劝(radicals: 又,力) 说是没办法改变你的...(...The persuasion of others can’t change you...)” under morphological constraint, whereas vanilla BERT produces an inferior correction “小(radicals: 小) 说 (fiction)”.

We also manually analyze the error cases of our model on the test datasets and find there are two common types of errors. One type is continuous errors, where several continuous characters in a sentence are wrong. For example, in sentence “...他们有时候，有一点捞到...”, “捞到(Caught)” are continuous errors, which should be “唠叨” (The correct sentence means ’Sometimes they are a little nagging’). The model fails to correct such continuous errors since the meaning of the whole sentence is more disturbed. Correcting another type of errors requires strong external knowledge. For instance, “心智 (mind)” in poem “...天将降大任于斯人也，必先苦其心智，劳其筋骨... (...When Heaven is going to give a great responsibility to someone, it will first fill his mind with suffering, toil his sinews and bones...)” is erroneous but semantic plausible in Chinese. The model is still unable to correct it into “心志 (mind)” since the model lacks knowledge of poem.

5 Conclusion

In this research, we propose a novel end-to-end trainable model called PHMOSpell for CSC, which incorporates both phonological and morphological knowledge from two feature extractors into a pretrained language model by an effective adaptive gating mechanism. Extensive experiments and empirical comparisons show that PHMOSpell achieves state-of-the-art results on three widely used benchmarks for CSC, demonstrating the effectiveness of the proposed method.

We remain extending the multi-modal knowledge to other NLP tasks (e.g., grammar error correction) as our future work. Another fruitful future work is exploring the integration of external knowledge so that the model can deal with errors in poems, proverbs, etc.

References

Zuyi Bao, Chen Li, and Rui Wang. 2020. Chunk-based chinese spelling check with global optimization. In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing: Findings, pages 2031–2040.

Eric Brill and Robert Moore. 2008. Spell checker with arbitrary length string-to-string transformations to improve noisy channel spelling correction. US Patent 7,366,983.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 286–293.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of language modeling for chinese spelling check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83.

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.

Hsun-Wen Chiu, Jian-Cheng Wu, and Jason S Chang. 2014. Chinese spell checking based on noisy channel model. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 202–209.

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.

Zijia Han, Chengguo Lv, Qiansheng Wang, and Guohong Fu. 2019. Chinese spelling check based on sequence labeling. In *2019 International Conference on Asian Language Processing (IALP)*, pages 373–378. IEEE.

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

Yu-Ming Hsieh, Ming-Hong Bai, and Keh-Jiann Chen. 2013. Introduction to ckip chinese spelling check system for sighan bakeoff 2013 evaluation. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 59–63.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph model for chinese spell checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92.

Ying Jiang, Tong Wang, Tao Lin, Fangjie Wang, Wenting Cheng, Xiaofei Liu, Chenghui Wang, and Weijian Zhang. 2012. A rule based chinese spelling and grammar detection system utility. In *2012 International Conference on System Science and Engineering (ICSSE)*, pages 437–440. IEEE.

Peng Jin, Xingyuan Chen, Zhaoyi Guo, and Pengyuan Liu. 2014. Integrating pinyin to improve spelling errors detection for chinese language. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 455–458. IEEE.

Karen Kukich. 1992. Techniques for automatically correcting words in text. *Acm Computing Surveys (CSUR)*, 24(4):377–439.

C-L Liu, M-H Lai, K-W Tien, Y-H Chuang, S-H Wu, and C-Y Lee. 2011. Visually and phonologically similar characters in incorrect chinese words: Analyses, identification, and applications. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(2):1–39.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A hybrid chinese spelling correction using language model and statistical machine translation with reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Lidia Mangu and Eric Brill. 1997. Automatic rule acquisition for spelling correction. In *ICML*, volume 97, pages 187–194. Citeseer.

Yuxian Meng, Wei Wu, Fei Wang, Xiaoya Li, Ping Nie, Fan Yin, Muyu Li, Qinghong Han, Xiaofei Sun, and Jiwei Li. 2019. Glyce: Glyph-vectors for chinese character representations. In *Advances in Neural Information Processing Systems*, pages 2746–2757.

Minh Nguyen, Gia H Ngo, and Nancy F Chen. 2020. Adaptable filtering using hierarchical embeddings for chinese spell check. *arXiv preprint arXiv:2008.12281*.

Hatem M Noaman, Shahenda S Sarhan, and M Rashwan. 2016. Automatic arabic spelling errors detection and correction based on confusion matrix-noisy channel hybrid system. *Egypt Comput Sci J*, 40(2):2016.

Arijit Sehanobish and Chan Hee Song. 2019. Using chinese glyphs for named entity recognition. *arXiv preprint arXiv:1909.09922*.

Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In

2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4779–4783. IEEE.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

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 the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37.

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. 2019a. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5780–5785.

Hao Wang, Bing Wang, Jianyong Duan, and Jiajun Zhang. 2019b. Chinese spelling error detection using a fusion lattice lstm. *arXiv preprint arXiv:1911.10750*.

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.

Weijian Xie, Peijie Huang, Xinrui Zhang, Kaiduo Hong, Qiang Huang, Bingzhou Chen, and Lei Huang. 2015. Chinese spelling check system based on n-gram model. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 128–136.

Jinhua Xiong, Qiao Zhang, Shuiyuan Zhang, Jianpeng Hou, and Xueqi Cheng. 2015. Hanspeller: a unified framework for chinese spelling correction. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 20, Number 1, June 2015-Special Issue on Chinese as a Foreign Language*.

Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 895–906.

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.

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.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.