SpellGCN: Incorporating Phonological and Visual Similarities into Language Models for Chinese Spelling Check

Xingyi Cheng* **Weidi Xu*** **Kunlong Chen**
Shaohua Jiang **Feng Wang** **Taifeng Wang** **Wei Chu** **Yuan Qi**
Ant Financial Services Group
{fanyin.cxy,weidi.xwd,kunlong.ckl,shaohua.jsh,zifan.wf,
taifeng.wang,weichu.cw,yuan.qi}@alibaba-inc.com

Abstract

Chinese Spelling Check (CSC) is a task to detect and correct spelling errors in Chinese natural language. Existing methods have made attempts to incorporate the similarity knowledge between Chinese characters. However, they take the similarity knowledge as either an external input resource or just heuristic rules. This paper proposes to incorporate phonological and visual similarity knowledge into language models for CSC via a specialized graph convolutional network (SpellGCN). The model builds a graph over the characters, and SpellGCN is learned to map this graph into a set of inter-dependent character classifiers. These classifiers are applied to the representations extracted by another network, such as BERT, enabling the whole network to be end-to-end trainable. Experiments [1] are conducted on three human-annotated datasets. Our method achieves superior performance against previous models by a large margin.

1 Introduction

Spelling errors are common in our daily life, caused typically by human writing, automatic speech recognition, and optical character recognition systems. Among these errors, misspelling a character frequently occurs due to the similarity between characters. In Chinese, many characters are phonologically and visually similar, but semantically very different. According to Liu et al. (2010), about 83% of errors are related to phonological similarity and 48% are related to visual similarity. The Chinese Spelling Check (CSC) task aims to detect and correct such misuse of the Chinese language. Despite recent development, CSC remains a challenging task. Notably, the spelling checking on Chinese is very different from English, due to its language nature. Chinese is a language consisting of many pictographic characters without word delimiters. And the meaning of each character changes dramatically when the context changes. Therefore, a CSC system needs to recognize the semantics and aggregate the surrounding information for necessary modifications.

Input (phonics)	餐厅的换经费产适合约会 cān tīng dē huàn jīng fèi chán shì hé yuē huì
BERT (phonics)	餐厅的月消费最适合约会 cān tīng dē yuè xiāo fèi zuì shì hé yuē huì
+SpellGCN (phonics)	餐厅的环境非常适合约会 cān tīng dē huán jìng fēi cháng shì hé yuē huì

Table 1: A CSC data sample from SIGHAN 2014 (Yu et al., 2014) with ID B1-3440-2, the incorrect/correct characters are in orange/blue. A BERT model modifies the text into a sentence that is semantically reasonable but dissimilar in pronunciation. By incorporating both phonological and visual similarities, our new method SpellGCN can generate a sentence that is both semantically sensible and phonically similar to the original sentence. The sentence output from SpellGCN means "this restaurant is very suitable for dating".

Previous methods followed the line of generative models. They used either language models (Liu et al., 2013, 2010; Yu and Li, 2014) or sequence-to-sequence models (Wang et al., 2019). To fuse the external knowledge of the similarity between characters, some of them leveraged a confusion set, which contains a set of similar character pairs. For instance, Yu and Li (2014) proposed to produce several candidates by retrieving the confusion set and then filter them via language models. Wang et al. (2019) used a pointer network to copy a similar character from the confusion set. These methods attempted to utilize the similarity information to confine the candidates, rather than modeling the relationship between characters explicitly.

In this paper, we propose a novel spelling check

*Equal contribution.

[1]The dataset and all code for this paper is available at https://github.com/ACL2020SpellGCN/SpellGCN

convolutional graph network (SpellGCN) that captures the pronunciation/shape similarity and explore the prior dependencies between characters. Specifically, two similarity graphs are constructed for the pronunciation and shape relationship correspondingly. SpellGCN takes the graphs as input and generates for each character a vector representation after the interaction between similar characters. These representations are then constructed into a character classifier for the semantic representation extracted from another backbone module. We use BERT (Devlin et al., 2019) due to its powerful semantic capacity. Combining the graph representations with BERT, SpellGCN can leverage the similarity knowledge and generate the right corrections accordingly. Regarding the example as in Table 1, SpellGCN is able to modify the sentence correctly within the pronunciation constraint.

Experiments were conducted on three open benchmarks. The results demonstrate that SpellGCN improves BERT evidently, outperforming all competitor models by a large margin.

In summary, our contributions are as follows:

- We propose a novel end-to-end trainable SpellGCN to integrate the pronunciation and shape similarities into the semantic space. Its essential components such as the specialized graph convolution and attentive combination operations are carefully investigated.
- We investigate the performance of SpellGCN both quantitatively and qualitatively. Experimental results indicate that our method achieves the best results on three benchmark datasets.

2 Related Work

The CSC task is a long-standing problem and has attracted much attention from the community. The research emerges in recent years (Jia et al., 2013; Xin et al., 2014; Yu and Li, 2014; Tseng et al., 2015; Fung et al., 2017; Wang et al., 2019; Hong et al., 2019), together with other topics, e.g., grammar error correction (GEC) (Rao et al., 2018; Ji et al., 2017; Chollampatt et al., 2016; Ge et al., 2018). CSC focuses on detecting and correcting character errors, while GEC also includes errors that need deletion and insertion. Previous work handles CSC using unsupervised language models (Liu et al., 2013; Yu and Li, 2014). The errors are detected/corrected by evaluating the perplexity of sentences/phrases. However, these models were unable to condition the correction on the input sentence. To circumvent this problem, several discriminative sequence tagging methods were adopted for CSC (Wang et al., 2018). For more flexibility and better performance, several sequence-to-sequence models were also employed (Wang et al., 2019; Ji et al., 2017; Chollampatt et al., 2016; Ge et al., 2018), as well as BERT (Hong et al., 2019).

Recent attention was paid to utilizing the external knowledge of character similarity. The similarity knowledge can be gathered into a dictionary, i.e., confusion set, where similar pairs are stored. Yu and Li (2014) first used the dictionary to retrieve similar candidates for potential errors. Wang et al. (2019) incorporated a copy mechanism into a recurrent neural model. When given similar characters as input, their model uses the copy mechanism to directly copy the character to the target sentence. In a sense, these models face difficulty in modeling the relationship between similar characters as the similarity information is solely used for candidate selection. To capture the pronunciation/shape similarity and explore the prior dependencies between characters, we propose to use graph convolution network (GCN) (Kipf and Welling, 2017) to model character inter-dependence, which is combined with the pre-training of BERT (Devlin et al., 2019; Cheng et al., 2019) for the CSC task.

GCN has been applied to model the relationship on several tasks. Yan et al. (2019) equipped it into the relation extraction task where relations construct a hierarchical tree. Li et al. (2018); Cheng et al. (2018) use it to model spatial-temporal to predict traffic flow. GCN was also used to model the relationship between labels in a multi-label task (Chen et al., 2019). In this paper, it is the first time that GCN is applied successfully into the CSC task. The relationship in CSC is much different from those tasks where objects in the graph are semantically related. By contrast, the similar characters are semantically distinct in CSC. Therefore, we deeply investigate the effect of our SpellGCN and propose several essential techniques.

3 Approach

In this section, we elaborate on our method for CSC. Firstly, the problem formulation is presented. Then, we introduce the motivations for SpellGCN, followed by its detailed description. At last, we present its application in the CSC task.

3.1 Problem Formulation

The Chinese Spelling Check task aims to detect and correct the errors in the Chinese language. When given a text sequence $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ consisting of n characters, the model takes $\mathbf{X}$ as input and output a target character sequence $\mathbf{Y} = \{y_1, y_2, ..., y_n\}$. We formulate the task as a conditional generation problem by modeling and maximizing the conditional probability $p(\mathbf{Y}|\mathbf{X})$.

3.2 Motivations

The framework of the proposed method is depicted in Figure 1. It consists of two components, i.e., a character representation extractor and a SpellGCN. The extractor derives a representation vector for each character. Above the extractor, SpellGCN is used to model the inter-dependence between characters. It outputs target vectors containing the information of similar characters after interactions.

As illustrated in Table 1, a vanilla language model is able to provide feasible corrections in semantic meaning but faces the difficulty in meeting the pronunciation constraint. Although the correction "月消费最" is semantically plausible, its phonics differs much from "换经费产" and "环境非常". This indicates that the similarity information between characters is necessary so that the model can learn to generate related answers. Previous methods have taken the similarity into consideration. However, they typically regarded similar characters as potential candidates, neglecting their inter-relationship in terms of pronunciation and shape. This work makes a preliminary attempt to handle this issue, trying to fuse both the *symbolic space* (phonological and visual similarity knowledge) and the *semantic space* (language semantic knowledge) into one model. To achieve this, we leverage the power of graph neural network (GNN) to infuse the similarity knowledge directly. The essential idea is to update the representations by aggregating the information between similar characters. Intuitively, a model is likely to have a sense of similar symbols when equipped with our method.

Among various GNN models, we use GCN in our implementation. Since there are up to 5K Chinese characters in the graph, the light-weight GCN is more suitable for our problem. The proposed SpellGCN is depicted as follows in detail.

3.3 Structure of SpellGCN

SpellGCN requires two similarity graphs $\mathbf{A}^p$, $\mathbf{A}^s$ for pronunciation and shape similarities correspondingly, which are derived from an open-sourced confusion set (Wu et al., 2013). For simplicity, the superscript will be omitted if unnecessary and $\mathbf{A}$ denotes one of these two similarity graphs. Each similarity graph is a binary adjacent matrix of size $\mathbb{R}^{N\times N}$, constructed from N characters in the confusion set. The edge $\mathbf{A}_{i,j} \in \{0, 1\}$ between i-th character and j-th character denotes whether the (i, j) pair exists in the confusion set.

The goal of SpellGCN is to learn a map function to map the input node embedding $\mathbf{H}^l \in \mathbb{R}^{N\times D}$ of l-th layer (where D is the dimensionality of character embedding) to a new representation $\mathbf{H}^{l+1}$ via convolutional operation defined by $\mathbf{A}$. This map function has two main sub-components: a graph convolution operation and an attentive graph combination operation.

Graph Convolution Operation The graph convolution operation is to absorb the information from neighboring characters in the graph. In each layer, the light-weight convolution layer in GCN (Kipf and Welling, 2017) is adopted:

$$f(\mathbf{A}, \mathbf{H}^l) = \hat{\mathbf{A}}\mathbf{H}^l\mathbf{W}_g^l, \quad (1)$$

where $\mathbf{W}_g^l \in \mathbb{R}^{D\times D}$ is a trainable matrix and $\hat{\mathbf{A}} \in \mathbb{R}^{N\times N}$ is the normalized version of the adjacent matrix $\mathbf{A}$. For the definition of $\hat{\mathbf{A}}$, we direct you to the original paper (Kipf and Welling, 2017). Note that we use the character embedding of BERT as the initial node features $\mathbf{H}^0$, and we omit the non-linearity function after convolution. Since we adopted BERT as our extractor, which has its own learned semantic space, we remove the activation function from the equation to keep the derived representation identical with original space, rather than a completely different space. During our experiments, using non-linearity activation such as ReLU is ineffective, resulting in a performance drop.

Attentive Graph Combination Operation The graph convolution operation handles the similarity of a single graph. To combine the pronunciation and shape similarity graphs, the attention mechanism (Bahdanau et al., 2015) is adopted. For each character, we represent the combination operation

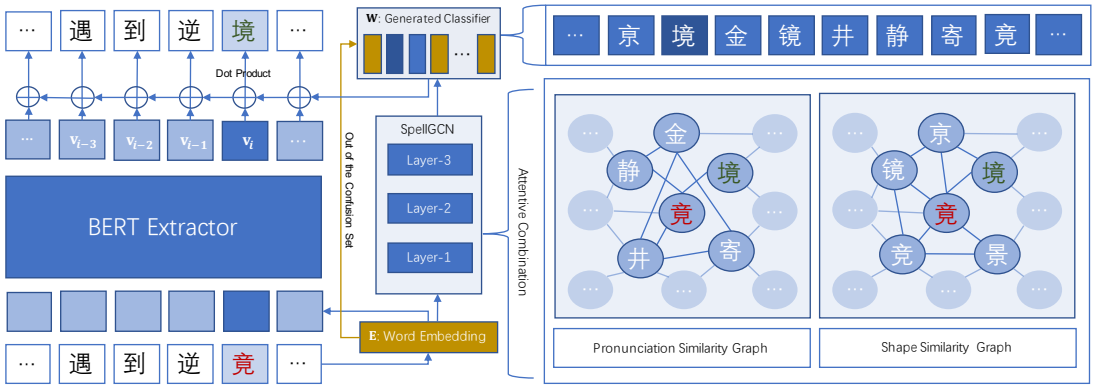


Figure 1: The framework of the proposed SpellGCN. **Left**: The characters in the input sentence are processed by the extractor to obtain the semantic representation vectors. **Right**: The phonological or visual similarity knowledge of characters is learned by our SpellGCN. Two similarity graphs are used to model the pronunciation and shape similarities respectively, and they are combined via an attentive combination operation. **Middle**: The character embedding vectors derived from SpellGCN are used as the target character classifiers .

as follows:

$$\mathbf{C}_i^l = \sum_{k \in \{s,p\}} \alpha_{i,k}^l f_k(\mathbf{A}^k, \mathbf{H}^l)_i \,, \tag{2}$$

where $\mathbf{C}^l \in \mathbb{R}^{N \times D}$ and $f_k(\mathbf{A}^k, \mathbf{H}^l)_i$ is the i-th row of convolved representation of graph k, $\alpha_{i,k}$ is a scalar for i-th character denoting the weight of graph k. The weight $\alpha_{i,k}$ is computed by

$$\alpha_{i,k} = \frac{\exp(\mathbf{w}_a f_k(\mathbf{A}^k, \mathbf{H}^l)_i / \beta)}{\sum_{k'} \exp(\mathbf{w}_a f_{k'}(\mathbf{A}^{k'}, \mathbf{H}^l)_i / \beta)}, \tag{3}$$

where $\mathbf{w}_a \in \mathbb{R}^D$ is a learnable vector shared across the layers and β is a hyper-parameter which controls the smoothness of attention weights. We found β essential for the attention mechanism.

Accumulated Output After graph convolution and attentive combination operations, we obtain a representation $\mathbf{C}^l$ for l-th layer. To maintain the original semantic of the extractor, all outputs of previous layers are accumulated as the output:

$$\mathbf{H}^{l+1} = \mathbf{C}^l + \sum_{i=0}^{l} \mathbf{H}^i \,. \tag{4}$$

In this way, SpellGCN is able to focus on capturing the knowledge of character similarity, leaving the responsibility of semantic reasoning to the extractor. Hopefully, each layer can learn to aggregate the information for the specific hop. During the experiments, the model failed when excluding $\mathbf{H}^0$.

3.4 SpellGCN for Chinese Spelling Check

Here, we introduce how to apply SpellGCN to the CSC task. Motivated by recent applications of GCN in relationship modeling (Chen et al., 2019; Yan et al., 2019), we use the final output of SpellGCN to be classifiers of the target characters.

Similarity Graphs from Confusion Set The similarity graphs used in SpellGCN are constructed from the confusion set provided in (Wu et al., 2013). It is a pre-defined set consisting of similar characters for most of (∼95%) the Chinese characters and these characters are categorized into five categories, i.e., (1) similar shape, (2) same pronunciation and same tone, (3) same pronunciation and different tone, (4) similar pronunciation and same tone, (5) similar pronunciation and different tone. Since the pronunciation similarity is more fine-grained compared with the shape similarity category, we combine the pronunciation similarities into one graph. Consequently, we construct two graphs corresponding to pronunciation and shape similarities.

Character Representation by Extractor The representation of characters used for final classification is given by an extractor. We can use any model that is able to output representation vectors $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\}$ (where $\mathbf{v}_i \in \mathbb{R}^D$) for n characters $\mathbf{X} = \{x_1, x_2, ..., x_n\}$. In our experiment, we adopt BERT as the backbone model. It takes $\mathbf{X}$ as input and uses the output of the last layer as $\mathbf{V}$. We conduct the experiment using the base version, which has 12 layers, 12 self-attention heads with a hidden size of 768 [2].

[2]This means D =768 in our experiment.

SpellGCN as Character Classifier When given the representation vector $\mathbf{v}_i$ of a character x_i, the model needs to predict a target character through a fully-connected layer whose weight $\mathbf{W} \in \mathbb{R}^{M \times D}$ is configured by the output of SpellGCN (M is the size of the extractor vocabulary):

$$p(\hat{y}_i|\mathbf{X}) = \text{softmax}(\mathbf{W}\mathbf{v}_i)\,. \tag{5}$$

Concretely, the output vectors of SpellGCN plays the role of the classifier in our task. We use the output of the last layer of SpellGCN $\mathbf{H}^L$ (where L is the number of layers) to classify the characters in the confusion set. And since the confusion set only covers a subset of vocabulary, we use the word embedding of the extractor as the classifier for those excluded by the confusion set. In this way, denoting $u_i \in \{1, ..., N\}$ is the index of confusion set for the i-th character in the extractor vocabulary, $\mathbf{W}$ is presented by:

$$\mathbf{W}_i = \begin{cases} \mathbf{H}^L_{u_i}, & \text{if } i\text{-th character} \in \text{confusion set} \\ \mathbf{E}_i, & \text{otherwise}\,, \end{cases} \tag{6}$$

where $\mathbf{E} \in \mathbb{R}^{M \times D}$ is the embedding matrix of extractor. In brief, we use the embedding from SpellGCN if the character is in the confusion set. Otherwise, the embedding vectors are used as in BERT. Instead of modeling a large compact graph containing all characters in the extractor vocabulary, we chose this implementation for computational efficiency, since there are around 5K characters in the confusion set and more than 20K characters in the extractor vocabulary.

Overall, the objective is to maximize the log likelihood of target characters:

$$\mathcal{L} = \sum_{\mathbf{X},\mathbf{Y}} \sum_i \log p(\hat{y}_i = y_i|\mathbf{X})\,. \tag{7}$$

3.5 Prediction Inference

The CSC task consists of two sub-tasks in evaluation, i.e., detection and correction. Some previous work (Yu and Li, 2014; Liu et al., 2013) used two models for these sub-tasks separately. In this work, we simply use the character with the highest probability $\arg\max_{\hat{y}_i} p(\hat{y}_i|\mathbf{X})$ as the prediction for the correction task. And the detection is achieved by checking whether the prediction matches the target character y_i.

4 Experiments

In this section, we describe our experiment in detail. We first present the training data and test data, as well as the evaluation metrics. Then we introduce our main results for SpellGCN. After that, the ablation studies are made to analyze the effect of the proposed components, followed by a case study. Finally, quantitative results are provided.

Training Data	# Line	Avg. Length	# Errors
(Wang et al., 2018)	271,329	44.4	382,704
SIGHAN 2013	350	49.2	350
SIGHAN 2014	6,526	49.7	10,087
SIGHAN 2015	3,174	30.0	4,237
Total	281,379	44.4	397,378

Test Data	# Line	Avg. Length	# Errors
SIGHAN 2013	1000(1000)	74.1	1,227
SIGHAN 2014	1062(526)	50.1	782
SIGHAN 2015	1100(550)	30.5	715

Graph	# Character	# Edges
Pronunciation Similarity Graph	4753	112,687
Shape Similarity Graph	4738	115,561

Table 2: Statistics information of the used data resources. The number in the bracket in #Line column denotes the number of sentences with errors.

4.1 Datasets

Training Data The training data is composed of three training datasets (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015), which has 10K data samples in total. Following (Wang et al., 2019), we also include additional 271K samples as the training data, which are generated by an automatic method (Wang et al., 2018) [3].

Test Data To evaluate the performance of the proposed method, we used three test datasets from the SIGHAN 2013, SIGHAN 2014, SIGHAN 2015 benchmarks (Wu et al., 2013; Yu et al., 2014; Tseng et al., 2015) as in (Wang et al., 2019). We also follow the same data pre-processing procedure, i.e., the characters in these datasets are converted to simplified Chinese using OpenCC [4]. The statistic of the data is listed in Table 2.

Baseline Models We compare our method with five typical baselines.

- LMC (Xie et al., 2015): This method utilizes the confusion set to replace the characters and then evaluates the modified sentence via a N-gram **L**anguage **M**odel.

[3] https://github.com/wdimmy/Automatic-Corpus-Generation

[4] https://github.com/BYVoid/

	Character-level						Sentence-level					
	Detection-level			Correction-level			Detection-level			Correction-level		
SIGHAN 2013	D-P	D-R	D-F	C-P	C-R	C-F	D-P	D-R	D-F	C-P	C-R	C-F
LMC (Xie et al., 2015)	79.8	50.0	61.5	77.6	22.7	35.1	(-)	(-)	(-)	(-)	(-)	(-)
SL (Wang et al., 2018)	54.0	69.3	60.7	(-)	(-)	52.1	(-)	(-)	(-)	(-)	(-)	(-)
PN (Wang et al., 2019)	56.8	91.4	70.1	79.7	59.4	68.1	(-)	(-)	(-)	(-)	(-)	(-)
FASpell (Hong et al., 2019)	(-)	(-)	(-)	(-)	(-)	(-)	76.2	63.2	69.1	73.1	60.5	66.2
BERT	80.6	88.4	84.3	98.1	87.2	92.3	79.0	72.8	75.8	77.7	71.6	74.6
SpellGCN	**82.6**	**88.9**	**85.7**	**98.4**	**88.4**	**93.1**	**80.1**	**74.4**	**77.2**	**78.3**	**72.7**	**75.4**
SIGHAN 2014	D-P	D-R	D-F	C-P	C-R	C-F	D-P	D-R	D-F	C-P	C-R	C-F
LMC (Xie et al., 2015)	56.4	34.8	43.0	71.1	50.2	58.8	(-)	(-)	(-)	(-)	(-)	(-)
SL (Wang et al., 2018)	51.9	66.2	58.2	(-)	(-)	56.1	(-)	(-)	(-)	(-)	(-)	(-)
PN (Wang et al., 2019)	63.2	82.5	71.6	79.3	68.9	73.7	(-)	(-)	(-)	(-)	(-)	(-)
FASpell (Hong et al., 2019)	(-)	(-)	(-)	(-)	(-)	(-)	61.0	53.5	57.0	59.4	52.0	55.4
BERT	82.9	77.6	80.2	96.8	75.2	84.6	**65.6**	68.1	66.8	**63.1**	65.5	64.3
SpellGCN	**83.6**	**78.6**	**81.0**	**97.2**	**76.4**	**85.5**	65.1	**69.5**	**67.2**	**63.1**	**67.2**	**65.3**
SIGHAN 2015	D-P	D-R	D-F	C-P	C-R	C-F	D-P	D-R	D-F	C-P	C-R	C-F
LMC (Xie et al., 2015)	83.8	26.2	40.0	71.1	50.2	58.8	(-)	(-)	(-)	(-)	(-)	(-)
SL (Wang et al., 2018)	56.6	69.4	62.3	(-)	(-)	57.1	(-)	(-)	(-)	(-)	(-)	(-)
PN (Wang et al., 2019)	66.8	73.1	69.8	71.5	59.5	69.9	(-)	(-)	(-)	(-)	(-)	(-)
FASpell (Hong et al., 2019)	(-)	(-)	(-)	(-)	(-)	(-)	67.6	60.0	63.5	66.6	59.1	62.6
BERT	87.5	85.7	86.6	95.2	81.5	87.8	73.7	78.2	75.9	70.9	75.2	73.0
SpellGCN	**88.9**	**87.7**	**88.3**	**95.7**	**83.9**	**89.4**	**74.8**	**80.7**	**77.7**	**72.1**	**77.7**	**75.9**

Table 3: The performance of our method and baseline models (%). D, C denote the detection, correction, respectively. P, R, F denote the precision, recall and F1 score, respectively. The results of BERT are from our own implementation. Best results are in **bold**. We performed additional fine-tuning on SIGHAN13 for 6 epochs as the data distribution in SIGHAN13 differs from other datasets, e.g. "的", "得" and "地" are rarely distinguished.

- SL (Wang et al., 2018): This method proposes a pipeline where a **S**equence **L**abeling model is adopted for detection. The incorrect characters are marked as 1 (0 otherwise).
- PN (Wang et al., 2019): This method incorporates a **P**ointer **N**etwork to consider the extra candidates from the confusion set.
- FASpell (Hong et al., 2019): This model utilizes a specialized candidate selection method based on the similarity metric. This metric is measured using some empirical methods, e.g., edit distance, rather than a pre-defined confusion set.
- BERT (Devlin et al., 2019): The word embedding is used as the softmax layer on the top of BERT for the CSC task. We trained this model using the same setting, i.e., the comparable model w/o SpellGCN.

Evaluation Metrics The precision, recall and F1 scores are reported as the evaluation metrics, which are commonly used in the CSC tasks. These metrics are provided for the detection and correction sub-tasks. Besides the evaluation on the character level, we also report the sentence-level metrics on the detection and correction sub-tasks, which is more appealing for real-world applications. On the sentence level, we consider a sentence to be correctly annotated only if all errors in the sentence are corrected as in (Hong et al., 2019) [5]. On the character level, we calculate the metrics using the evaluation script from (Wang et al., 2019) [6]. We also evaluated BERT and SpellGCN by the official evaluation metrics tools[7], which gives False Positive Rate (FTR), Accuracy and Precision/Recall/F1.

4.2 Hyper-parameters

Our code is based on the repository of BERT [8]. We fine-tune the models using AdamW (Loshchilov and Hutter, 2018) optimizer for 6 epochs with a batch size of 32 and a learning rate of 5e-5. The number of the layer in SpellGCN is 2, and the attentive combination operation with factor 3 is used. All experiments were conducted for 4 runs and the averaged metric is reported. The code and trained models will be released publicly after review (currently, the code is attached in the supplementary files).

[5] https://github.com/iqiyi/FASPell

[6] https://github.com/wdimmy/Confusionset-guided-Pointer-Networks-for-Chinese-Spelling-Check

[7] http://nlp.ee.ncu.edu.tw/resource/csc.html

[8] https://github.com/google-research/bert

SIGHAN 2014	FPR	D-A	C-A	D-P	D-R	D-F	C-P	C-R	C-F
BERT	15.3	76.8	75.7	81.9	68.9	74.9	81.4	66.7	73.3
SpellGCN	**14.1**	**77.7**	**76.9**	**83.1**	**69.5**	**75.7**	**82.8**	**67.8**	**74.5**
SIGHAN 2015	FPR	D-A	C-A	D-P	D-R	D-F	C-P	C-R	C-F
BERT	13.6	83.0	81.5	85.9	78.9	82.3	**85.5**	75.8	80.5
SpellGCN	**13.2**	**83.7**	**82.2**	**85.9**	**80.6**	**83.1**	85.4	**77.6**	**81.3**

Table 4: The performance of BERT and SpellGCN evaluated by official tools on SIGHAN 2014 and SIGHAN 2015. FPR denotes the false positive rate and A denotes the accuracy. D-A and C-A denote detection accuracy and correction accuracy.

4.3 Main Results

Table 3 shows the performance of the proposed method on the three CSC datasets, compared with five typical CSC systems. When using SpellGCN, the model achieves better results in all test sets against vanilla BERT, which verifies its effectiveness. The improvement is considerable with such a large amount of training data (cf. the comparison in Figure 2). This indicates the similarity knowledge is essential for CSC and it can hardly be learned by simply increasing the data amount. In terms of sentence-level F1score metric in the correction subtask, i.e., C-F score in the last column, the improvements against previous best results (FASPell) are 9.2%, 9.7% and 13.3% respectively. Nevertheless, it should be noted that FASpell was trained on different training data while this paper follows the setting mentioned in the PN paper (Wang et al., 2019). Ideally, our method is compatible with FASpell and better results can be achieved when FASpell is employed.

FASpell used their own metrics, which are different from the sentence-level false postive and false negtivate counting strategy of the official evaluation toolkit. We used the scripts by PGNet and FASpell to compute their metrics for fair comparison. We further add the official evaluation results of BERT and SpellGCN in Table 4. Actually, SpellGCN consistently improves the performance when evaluated by the PGNet/FASpell scripts and the official evaluation toolkit. We will add the FPR results in our revision. The FPR scores are 14.1% (SpellGCN) v.s. 15.3% (BERT) on SIGHAN 14, and 13.2% (SpellGCN) v.s. 13.6% (BERT) on SIGHAN 15. FPR on SIGHAN 13 is statistically meaningless since almost all the tested sentences have the spelling errors.

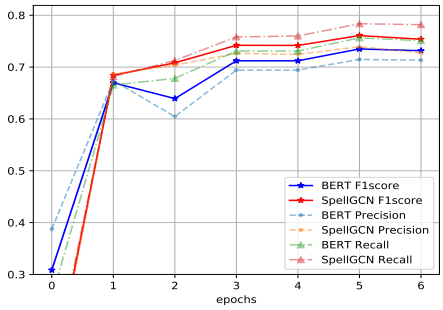


Figure 2: The test curves for sentence-level correction metrics with and without SpellGCN w.r.t. the number of training epochs on SIGHAN 2015.

4.4 Ablation Studies

In this subsection, we analyze the effect of several components, including the number of layers and the attention mechanism. The ablation experiments were performed using 10K training data.

Effect of the Number of Layers Generally, the performance of a GCN varies with the number of layers. We investigate how the number of SpellGCN layers influence the performance in CSC. In this comparison, the number of layers changes from 1 to 4, and the results are illustrated in Figure 3. For clarity, we report the character-level C-F on the three test datasets. The results indicate that SpellGCN is able to make use of multiple layers. With multiple layers, SpellGCN can aggregate the information in more hops and therefore, achieve better performance. However, the F1score drops when the number of layers is larger than 3. This is reasonable due to the over-smooth problem noted in (Yan et al., 2019). When the number of GCN layers increases, the representations of neighboring characters in the similarity graph will get more and more similar since they all are calculated via those of their neighbors in the similarity graph.

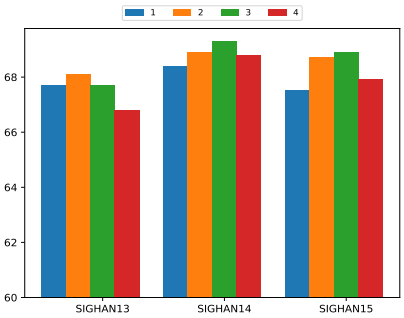


Figure 3: The character-level C-F results (%) w.r.t. the depth of SpellGCN. The results were obtained with 10K training samples.

combination method	C-F
baseline (w/o SpellGCN)	67.0
sum pooling	66.3
mean pooling	67.5
attentive combination (β=1)	67.8
attentive combination (β=3)	68.2
attentive combination (β=5)	68.0
attentive combination (β=10)	67.7

Table 5: The ablation results for graph combination method (%). The averaged character-level C-F scores of 4 runs on the SIGHAN 2013 are reported. The models were trained with 10K training samples. Mean pooling denotes that the output representation C^l of each layer is the average of $f_{k\in\{P,S\}}(\mathbf{A}_k, \mathbf{H}^l)$, while sum pooling summarizes $f_{k\in\{P,S\}}(\mathbf{A}_k, \mathbf{H}^l)$.

Effect of Attention Mechanism We investigate how to better combine the graphs in the SpellGCN layer. Here, we compare the attention mechanism against sum-pooling and mean-pooling, with different hyper-parameter β mentioned in Section 3.3. The experiments are conducted based on the 2-layer SpellGCN on SIGHAN 2013 test set. The results presented in Table 5 show that the sum pooling fails in the CSC task. We suggest that the sum pooling is inconsistent with the normalization of GCN and fails to combine the information from different channels (i.e., graphs). The mean pooling is feasible but is surpassed by the attention mechanism. This indicates that the adaptive combination for each character node is beneficial. We incorporate a hyper-parameter β into the attention operation since the dot products may grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. With these results, we chose the attention mechanism with a β of 3 in SpellGCN.

Pronunciation: fǎng→fán, wàng→wàng
...走路真的麻坊，我也没有喝的东西，在家汪了... ...走路真的麻木，我也没有喝的东西，在家呆了... ...走路真的麻烦，我也没有喝的东西，在家忘了...
Pronunciation: yīn→yǐng
...因为妈妈或爸爸在看录音机...帮小孩子解决问题... ...因为妈妈或爸爸在看录音机...帮小孩子解决问题... ...因为妈妈或爸爸在看录影机...帮小孩子解决问题...
Shape: 向→尚
...不过在许多传统文化的国家，女人向未得到平等... ...不过在许多传统文化的国家，女人从未得到平等... ...不过在许多传统文化的国家，女人尚未得到平等...

Table 6: Several prediction results on the test set. The first line in the block is the input sentence. The second line is corrected by BERT without SpellGCN. And the last line is the result from SpellGCN. We highlight the incorrect/correct characters by orange/blue color.

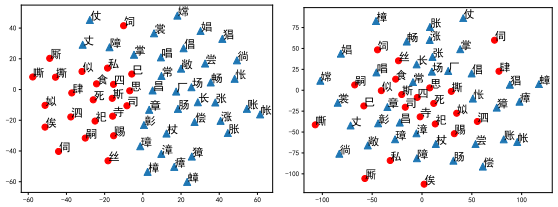
4.5 Case Study

We show several correction results to demonstrate the properties of SpellGCN. In addition to the sample illustrated in Table 1, several prediction results are given in Table 6. From these cases, we can tell that our SpellGCN is capable of revising the incorrect characters into correct ones with the pronunciation and shape constraint. For instance, in the first case, "麻坊(fǎng)" is detected as errors and modified into "麻烦(fán)". Without pronunciation similarity constraint, "麻木(mù)" becomes the most probable answer. And surprisingly, in the second case, our SpellGCN successfully modifies the character reasonable in the context. The meaning of input sentence "看录音机" is "watch the audio recorder", and our method corrects it into "看录影机" which means "watch the video recorder". We suggest that SpellGCN injects a prior similarity between "音" and "影" in the representation space so that the model derives a higher posterior probability of "影". In the last case, we show a correction result under the shape constraint. In the confusion set, "向" is similar to "尚" and therefore, using SpellGCN is able to retrieve the correct result.

4.6 Character Embedding Visualization

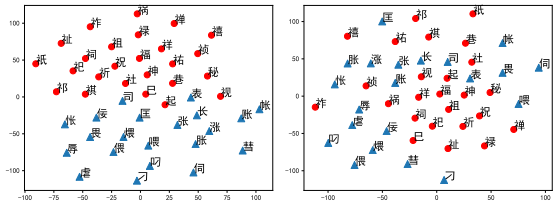
Previous experiments have explored the performance of SpellGCN quantitatively in detail. To qualitatively study whether SpellGCN learns meaningful representations, we dive into the target embedding space $\mathbf{W}$ derived from SpellGCN.

In Figure 4, the embedding of characters with phonics "cháng" and "sì" is presented using t-SNE (Maaten and Hinton, 2008). The embedding



(a) w/ SpellGCN (b) w/o SpellGCN

Figure 4: The scatter of similar characters of ”长” and “祀” in terms of pronunciation by t-SNE.



(a) w/ SpellGCN (b) w/o SpellGCN

Figure 5: The scatter of similar characters of ”长” and “祀” in terms of shape by t-SNE.

learned by BERT captures the semantic similarity but fails to model the similarity in terms of pronunciation for the CSC task. This is reasonable as this similarity knowledge is absent in the modeling. In contrast, our SpellGCN successfully infuses this prior knowledge into the embedding and the resulting embedding exhibits cluster patterns. The embedding of characters with these two different pronunciations forms two clusters, corresponding to “cháng” and “sì” respectively. Due to this property, the model tends to recognize similar characters and hence is able to retrieve the answers under pronunciation constraint. Figure 5 shows the same situation for the shape similarity, where two sets of characters with the shape similar to “长” and “祀” are scattered. This verifies the ability of SpellGCN in modeling shape similarity.

5 Conclusions

We proposed SpellGCN for CSC to incorporate both phonological and visual similarities into language models. The empirical comparison and the results of analytical experiments verify its effectiveness. Beyond CSC, SpellGCN can be generalized to other situations where specific prior knowledge is available, and to other languages by leveraging specific similarity graphs analogously. Our method can also be adapted to grammar error correction, which needs insertion and deletion, by utilizing more flexible extractors such as Levenshtein Transformer (Gu et al., 2019). We leave this direction to future work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-label image recognition with graph convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 5177–5186.

Xingyi Cheng, Weidi Xu, Kunlong Chen, Wei Wang, Bin Bi, Ming Yan, Chen Wu, Luo Si, Wei Chu, and Taifeng Wang. 2019. Symmetric regularization based BERT for pair-wise semantic reasoning. *CoRR*, abs/1909.03405.

Xingyi Cheng, Ruiqing Zhang, Jie Zhou, and Wei Xu. 2018. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pages 1–8. IEEE.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2768–2774.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Gabriel Pui Cheong Fung, Maxime Debosschere, Dingmin Wang, Bo Li, Jia Zhu, and Kam-Fai Wong. 2017. NLPTEA 2017 shared task - chinese spelling check. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications, NLP-TEA@IJCNLP 2017, Taipei, Taiwan, December 1, 2017*, pages 29–34.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1055–1065.

Jiatao Gu, Changhan Wang, and Jake Zhao. 2019. Levenshtein transformer. *CoRR*, abs/1905.11006.

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.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 753–762.

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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 88–92.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Jing Li, Hao Peng, Lin Liu, Guixi Xiong, Bowen Du, Hongyuan Ma, Lihong Wang, and Md. Zakirul Alam Bhuiyan. 2018. Graph cnns for urban traffic passenger flows prediction. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2018, Guangzhou, China, October 8-12, 2018*, pages 29–36. IEEE.

Chao-Lin Liu, Min-Hua Lai, Yi-Hsuan Chuang, and Chia-Ying Lee. 2010. Visually and phonologically similar characters in incorrect simplified chinese words. In *COLING 2010, 23rd International Conference on Computational Linguistics, Posters Volume, 23-27 August 2010, Beijing, China*, pages 739–747.

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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 54–58.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Gaoqi Rao, Qi Gong, Baolin Zhang, and Endong Xun. 2018. Overview of NLPTEA-2018 share task chinese grammatical error diagnosis. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications, NLPTEA@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 42–51.

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, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, 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, Brussels, Belgium, October 31 - November 4, 2018*, pages 2517–2527.

Dingmin Wang, Yi Tay, and Li Zhong. 2019. Confusionset-guided pointer networks for chinese spelling check. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5780–5785.

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, SIGHAN@IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, 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, SIGHAN@IJCNLP 2015, Beijing, China, July 30-31, 2015*, pages 128–136.

Yang Xin, Hai Zhao, Yuzhu Wang, and Zhongye Jia. 2014. An improved graph model for chinese spell checking. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing, Wuhan, China, October 20-21, 2014*, pages 157–166.

Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5770–5774.

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, Wuhan, China, October 20-21, 2014, 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, Wuhan, China, October 20-21, 2014*, pages 126–132.