Automatic Proofreading in Chinese: Detect and Correct Spelling Errors in Character-level with Deep Neural Networks

Qiufeng Wang[1,2], Minghuan Liu[1,2], Weijia Zhang[1,2], Yuhang Guo[1,2], and Tianrui Li[1,2]

[1] School of Information Science and Technology, Southwest Jiaotong university, 999 Xi'an Road, Chengdu, China
[2] {qfwang,ericliu,wjzhang,guoyuhang,trli}@my.swjtu.edu.cn

Abstract. Rapid increase of the scale of text carries huge costs for manual proofreading. In comparision, automatic proofreading shows great advantages on time and human resource, drawing more researchers into it. In this paper, we propose two attention based deep neural network models combined with confusion sets to detect and correct possible Chinese spelling errors in character-level. Our proposed approaches first model the context of Chinese character embedding using Long Short-Term Memory (LSTM) networks, then score the probabilities of candidates from its confusion set through attention mechanism, choosing the highest one as the prediction answer. Also, we define a new methodology for obtaining (preceding text, following text, candidates, target) quads and provides a supervised dataset for training and testing[3]. Performance evaluation indicates that our models achieve the state-of-the-art performance and outperform a set of baselines.

Keywords: Error detection of Chinese text · Error correction of Chinese text · LSTM model · Attention mechanism.

1 Introduction

The increasing scale of text carries a huge cost for manual proofreading, calling for necessary and meaningful automatic text proofreading, which is to automatically detect and correct the errors in pieces of text with machine. Text proofreading contains spelling check, grammar check, punctuation check, digit check, and others. Most of current works including ours focus on spelling check, which is an important preprocessing task since spelling errors appear in every written language. For instance, given one sentence of text:

Ex.1-right 今天的天气很好。

Ex.1-right *The weather is good today.*

However, due to some reasons, OCR identification error, for example, misrecognize "今(jin)" as "令(ling)", then it becomes:

Ex.1-wrong 令天的天气很好。

[3] Our data has been released to the public in https://github.com/ccit-proofread .

The sentence above shows a spelling error, in which "令(make, order, pinyin:*ling*)" is a wrongly written character. Our task aims to detect it and correct it properly into "今(this, now, pinyin:*jin)*" while taking no action on the other characters. Therefore, spelling check generally contains two steps, e.g. detection and correction[14]. Detection identifies errors and correction corrects them.

Many challenges are required to be solved for proofreading Chinese spelling errors. (i) Chinese words have no obvious boundary. So many works [14] take it as a prerequisite to introduce automatic word segmentation although segmentation itself may give rise to errors, and ultimately affects the whole performance. (ii) The number of Chinese charaters is large and they own similarities. *Grand Chinese Characters Dictionary* collects more than 56,000 Chinese characters. Besides, these Chinese characters have a lot of similarities like similar pronunciation, similar shape and similar meaning [3] . (iii) A single model always behaves variously in different error ratio text.

In this work, we propose two novel attention based deep neural network models to address the above problems. We frame spelling check as a prediction problem, and introduce confusion sets which provides sets of possible candidate characters that are potential to be confused. To train the model, we introduce a new method to build a spelling check dataset. Simplified Chinese text collected from Wiki and Baidu containing no error are converted to (preceding text, following text, candidates, target) quads. Since the error rate of a qualified publishing book must be lower than 0.01%, we set four different proportions of errors like 0.1%, 0.5%, 1% and 2% to evaluate the proofreading performance in different situations. In the appendix, we elaborate on the production process of the dataset.

Our work includes three solutions to meet the above three challenges. (i) We avoid automatic segmentation errors by handling in character-level, which means we detect and correct character errors rather than words. (ii) We collect character-level confusion sets as priori knowledge to reduce the suggested correction candidates, which cover errors due to character similarities. (iii) We extract more global information using encoders to reduce the impact of local errors. Moreover, we introduce confidence threshold to make our model adjustable for different error ratio text.

2 Related Work

Automatic detection and correction in text is a significant task in NLP. Researches for English launch earlier and have developed maturely; however, study on Chinese begins late and still on exploration. In this section, we introduce related work for automatic text proofreading on spelling check in English and Chinese.

For English, Golding et al. [15] proposed Bayesian and Winnow algorithm based on machine learning methods, took spelling errors correction as words disambiguation by learning the features of words in confusion sets and comparing them with specific contextual feature to choose the most suitable word. Hirst et al. [16] used semantic information to detect real word errors by computing semantic distances between words and contexts with WordNet. If the distance is close, the word is right; otherwise is wrong. A weakness of this method is that there is no consideration of the relation between easily confused words.

For Chinese, Tseng et al. [11] proposed a bake-off for Chinese spelling check, leading a lot of works on this task. Many current works used word segmentation and language models based on large-scale corpus statistics to detect spelling errors in text. Zhang et al. [14] proposed an approximate Chinese word matching algorithm. With a Tri-gram language model, detection and correction were achieved at the same time. Zhuang et al. [18] extracted both syntax and semantic features from confusion sets of similar characters using N-gram and LSA language model. Liu et al. [4] scored words from provided confusion sets in specific context by N-gram model and found the highest one as results. Some researchers [17] applied hybrid approaches for detection and correction and achieved good results.

In our work, we incorporate attention mechanisms with confusion sets for spelling errors detection and correction. Attention mechanisms have been applied in various works, which allow models to draw deep dependencies between two parts. They were first raised in [7], then adopted by MT works [1], resolving alignment between the source and the corresponding target. Vaswani et al. [12] raised multi-head attention and self-attention, making attention mechanisms even more prevalent.

3 Models

In this section, a normal definition of our task and embedding of Chinese character is firstly given. Afterwards, brief introductions and modeling of LSTM networks are given. Lastly, our porposed deep neural network models based on attention mechanism for spelling errors detection and correction are presented.

3.1 Task definition and embedding

In this paper, we adopt deep neural networks to mine deep information of a sequence, modeling the preceding and following contexts of one specific character. Given a sequence $x = \{x_1, x_2, \cdots, x_{k-1}, x_k, x_{k+1}, \cdots, x_{n-1}, x_n\}$, where x_k the k-th character in one given sequence, n is the sequence length. We want to identify if x_k is wrong, then the confusion set $\mathcal{D}(x_k)$ of x_k is introduced. Our task is to estimate the probability of candidate d_k from the preceding context xp_k and the following context sequence xf_k as $P(d_k \mid xp_k, xf_k, \theta)$, where $d_k \in \mathcal{D}(x_k)$, $xp_k = \{x_1, x_2, \cdots, x_{k-1}\}$, $xf_k = \{x_{k+1}, x_{k+2}, \cdots, x_n\}$, θ is hyperparameter of Deep Neural Network. For data representation, we map all characters into vector representations using embedding fuction $\mathcal{E}$ with all vectors stacked in a embedding matrix $L \in \mathbb{R}^{h \times \|V\|}$, where h the dimension of vectors and V the vocabulary size.

3.2 Modeling with Long Short-Term Memory (LSTM)

LSTM has recently made considerable success in several NLP tasks, showing a remarkable ability to represent long sequences in vectors. LSTM is a great improvement of Recurrent Neural Network (RNN) [2], aiming to solve problems of gradient vanishing and exploding over long sequences for RNN.

For deep neural network models in NLP, an RNN Encoder architecture is always adopted for modeling text [9,5]. We believe RNN encoders are enough to extract information needed for our task. Inspired by Tang et al.[10], for one specific character to be checked, we utilize two standard LSTM networks, namely, LSTM$_p$ and LSTM$_f$ to model its preceding and following contexts, encoding which into fixed length vectors. For x_k, LSTM$_p$ encodes preceding context $\{x_1, x_2, \cdots, x_{k-1}\}$ into $\overrightarrow{h_{k-1}}$ while LSTM$_f$ encodes following context $\{x_{k+1}, x_{k+2}, \cdots, x_l\}$ reversely into $\overleftarrow{h_{k+1}}$. For the whole context the composite output vector h_k is obtained by concatenation:

$$h_k = \left[\overrightarrow{h_{k-1}}; \overleftarrow{h_{k+1}}\right] \tag{1}$$

In the next subsection we use the attention mechanism to further extract important information from h_k. The goal is to calculate the importance of the preceding and following context for predicting correct candidates.

3.3 Attention Based Neural Network Models

Attention mechanisms allow models to mine deep relation between two parts. A general attention function proposed by Luong et al. [5] computes context vectors c_t as follows:

$$r(h_t, h_s) = h_t^T W_a h_s \tag{2}$$

$$a_t(s) = softmax(r(h_t, h_s)) \tag{3}$$

$$c_t = \sum_s a_t(s) h_s \tag{4}$$

where $W_a \in \mathbb{R}^{\|h_t^T\| \times \|h_s\|}$, h_t the target hidden state, h_s the source hidden state, $r(h_t, h_s)$ the relation score and $a_t(s)$ the normalization score of h_t and h_s. Next we propose two models based on the attention mechanism:

The Candidate Attentive Checker

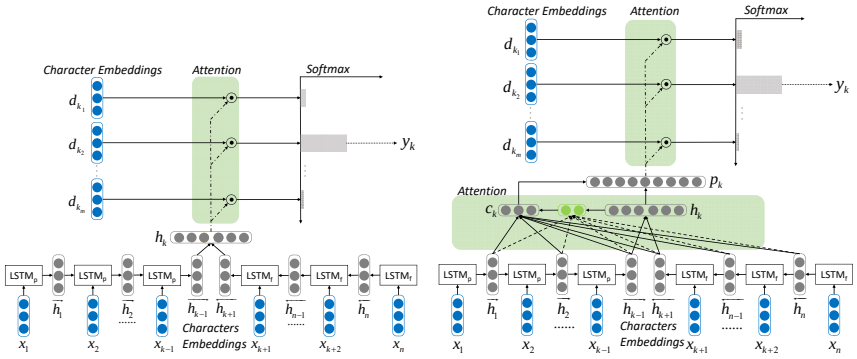
We try to adopt an attention function to mark each candidate in specific context. A normal thought computes candidate scores through c_t. Note that Eq.4 computes every c_t as a weighted average, and each h_s is computed from the same context, which is relevant with each other. However, candidates in confusion sets are independent, which means the sum operation in Eq.4 will instead impair the information. Hence we decide to directly calculate the score α_{k_i} of hidden state h_k obtained by concatenation and i-th candidate d_{k_i} without computing weighted average context vectors as:

$$\alpha_{k_i} = softmax(h_k^T W_s d_{k_i}) \tag{5}$$

where $W_s \in \mathbb{R}^{\|h_k^T\| \times \|d_{k_i}\|}$. In this type, α_k is a variable-length vector, whose size equals the number of candidates.

As shown in Fig.1(a), candidate attentive checker employs the above attention mechanism beyond the context vector and candidates. The model computes an attention score α_{k_i} of each i-th candidate d_{k_i} with respect to character x_k followed by 5. The model marks y_k with the highest score as the answer prediction.

$$y_k = argmax(\alpha_k) \tag{6}$$



(a) Candidate Attentive Checker (b) Double Attentive Checker

Fig. 1. Attention Based Neural Network Models

The Double Attentive Checker

The candidate attentive checker is able to focus on candidates that are most likely to fit for contextual features. Such context vectors concatenated only by the last hidden states of LSTM_p and LSTM_f are impacted more by closer characters, tending to cause long-distance dependencies loss. If these characters are wrong in the given sequence, it will result in bad effects on current prediction. Hence we go further by applying an general attention mechanism as Eqs. (2,3,4), illustrated in Fig. 1(b), to extract dependencies between each hidden state h_s of input sequece x and the context vector h_k. By doing such, double attentive checker pays more attention on correctly relevant characters without losing long-distance information. We compute the dependency representation vector p_k as follows:

$$r_{k_s} = h_k^T W_a h_s \tag{7}$$

$$\alpha_{k_s} = softmax(r_{k_s}) \tag{8}$$

$$c_k = \sum_s \alpha_{k_s} h_s \tag{9}$$

$$p_k = [c_k; h_k] \tag{10}$$

Ultimately, double attentive checker computes scores through attention similar as candidate attentive checker. y_k with the highest score is the prediction.

$$\alpha_{k_i} = softmax(p_k^T W_s d_{k_i}) \tag{11}$$

$$y_k = argmax\,(\alpha_k) \tag{12}$$

where $W_s \in \mathbb{R}^{3h \times h}$.

3.4 Model Training

Our models use the cross entropy as the objective function, and plus an L2 regularization term to prevent overfitting as follows:

$$loss = -\sum_{i=1}^{N} \sum_{d \in \mathcal{D}(x_i)} \delta_d(x_i) log(P_d(x_i; \theta)) + \lambda \|\theta\|^2 \tag{13}$$

where N the samples of training data, $\mathcal{D}(x_i)$ the confusion set of x_i, $\delta_d(x_i)$ indicates whether candidate d is the right answer of x_i, whose value is 1 or 0, θ is the parameter set, and λ is the regularization weight. . And $P_d(x_i)$ is the probability predicting x_i as d given by the attention score after the softmax layer.

3.5 Confidence Modification

Both Candidate Attentive Proofreader and Double Attentive Proofreader take the candidate with the highest score as the prediction answer. In order to adapt to text of different error scales and prevent these models from choosing an unreasonable answer, we propose to employ a confidence modification on our models. Given character x_k and the answer prediction y_k, we apply the following modification function to acquire the final answer z_k,

$$z_k = \begin{cases} y_k & \alpha_{k_{arg\, y_k}} \cdot t > \alpha_{k_{arg\, x_k}} \\ x_k & \textit{others} \end{cases} \tag{14}$$

where t is the confidence threshold parameter of recognizing the prediction answer. When a model scores y_k lower than the threshold, then it is not very confident about its choice and will keep the original character x_k. Through confidence modification, we keep these non-errors as much as possible and only choose answers with high confidence.

4 Experiments

We first describe our experimental setup, then evaluate our models by comparing with some baselines and report the empirical results in this section.

4.1 Experimental Setup

Training details We first set vocabulary size $\|V\|$ as 10K, which contains the top 10K characters ordered by frequency in Wiki, and map all Out of Vocabulary (OOV) characters to '<UNK>' tokens. We select embedding size h $[\underline{128}, 256, 512]$, and mini-batch size $[64, \underline{128}, 256]$. Embeddings are trained with the model. We consider adam update rule [8] with initial learning rate $[0.0005, \underline{0.001}, 0.01]$. We apply dropout with probability $[0.1, \underline{0.2}, 0.3, 0.4]$ to LSTM cells and embedding layer. We assign the threshhold parameter t in Eq.14 as 0.001. All experiments are run on single GPU, GeForce GTX

TITAN Xp with 12GB VRAM. We use our dataset for training and testing. Each training set and test set extract 200K sentences from the dataset. Note that test set is mixed with errors and we set four different ratios to test model performances, which include 0.1%, 0.5%, 1% and 2%.

Comparision Settings Since there is no such available benchmark approaches for this task, we compare with the following baseline methods:

MDMM (Mays, Damerau and Mercer Model): This statistical model was proposed by Mays et al. [6] which is based on Bayes theorem to tackle the english real-word correction. They regard this task as a noisy channel problem in which an intended statement undergoes introduction of typos and then the most likely statement are predicted by their system.

DAM (Dynamic Alpha Model): This is a method proposed by Hearn et al. [13] to address the problem in MDMM that there is no probability mass left to be distributed among the confusion set sometimes. They introduce a prior belief of parameters and modify the prior belief using fluctuation factors.

CSNG (Confusion sets based N-gram/language model): This method was previously proposed by Liu et al. [4] to detect and correct real-word errors in Chinese. It extracted adjacent bigrams and trigrams feature to compute the probability with the confusion words in the aim word' s confusion set, then scores those words with multi-features. We test this model on characters.

CAC (Candidate Attentive Checker) , **DAC** (Double Attentive Checker): Our proposed attention based deep model as discussed at Section. 3.3.

Evaluation Metrics To evaluate the proofreading performance of our models, followed by Tseng et al. [11], we select the widely-used measurements of Precision, Recall, F1 Score computed by characters both on detection-level and correction-level as our metrics.

4.2 Empirical Results

Table 1 and table 2 show the results of our models compared with baseline methods on test data of four different error ratios. It is clearly that our proposed models outperform all previous baseline methods on both detection-level and correction-level, which indicates that our models utilize more context information with sets of candidates than those who only focus on local features. In addition, on detection-level, DAM tends to emphasize the recall much more than precision, leading to bad effects on F1 scores; on the opposite, CAC and DAC both get better precisions and F1 scores. It should be noted that in this task, with the same F1 score, precision is usually more imporant than recall because the number of non-errors is always much larger than errors.

Furthermore, as the error ratio increases, recalls drop while precisions elevate, showing that these two models perform better on data containing more errors. Moreover, DAC performs better than CAC overall, but their differences are less evident with error ratio grows due to DAC extracting more useful global information using attention mechanism in less errors sentences, which is less possibly influenced by nearby errors.

	0.001			0.005			0.01			0.02			Overall		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
MDMM	0.064	0.409	0.110	0.254	0.403	0.312	0.399	0.392	0.396	0.567	0.385	0.459	0.321	0.398	0.319
DAM	0.027	**0.781**	0.052	0.125	**0.783**	0.216	0.221	**0.767**	0.343	0.363	**0.750**	0.489	0.184	**0.770**	0.275
CSNG	0.065	0.689	0.118	0.246	0.655	0.358	0.382	0.635	0.477	0.534	0.598	0.565	0.307	0.573	0.380
CAC	0.267	0.753	0.394	0.619	0.693	0.654	0.747	0.647	0.693	0.839	0.586	0.690	0.618	0.670	0.608
DAC	**0.296**	0.766	**0.427**	**0.651**	0.702	**0.676**	**0.771**	0.654	**0.708**	**0.856**	0.587	**0.696**	**0.644**	0.677	**0.627**

Table 1. Detection-level Performance Comparison

	0.001			0.005			0.01			0.02			Overall		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
MDMM	0.050	0.322	0.087	0.194	0.308	0.238	0.317	0.312	0.315	0.447	0.303	0.361	0.252	0.311	0.250
DAM	0.020	0.572	0.038	0.089	0.557	0.153	0.160	0.555	0.248	0.259	**0.534**	0.349	0.132	0.554	0.197
CSNG	0.062	0.656	0.113	0.229	0.610	0.333	0.346	**0.576**	0.432	0.464	0.519	0.490	0.275	0.590	0.342
CAC	0.252	0.711	0.372	0.563	0.631	0.595	0.657	0.569	0.610	0.700	0.488	0.575	0.543	0.600	0.538
DAC	**0.279**	**0.721**	**0.402**	**0.594**	**0.640**	**0.616**	**0.678**	0.575	**0.622**	**0.719**	0.493	**0.585**	**0.568**	**0.607**	**0.556**

Table 2. Correction-level Performance Comparison

In experiments on our dataset, these baselines get quitely worse performances especially precisions than results shown in their paper, but with the error ratio increasing, we get closer empirical results to theirs. So we guess that their experiments are conducted on data of relatively high error ratio, and their methods may be powerless on data with a low error ratio. In the appendix, we present some example answers from CAC and DAC.

4.3 Error Ratio Analysis

A good spelling checker should correct more errors than it introduces, hence we denote Error Difference = (#errors before checking - #errors after checking) / #errors before checking, and compare baselines with our models. In table 3, we show the comparision results. We can observe that on a very low error ratio as 0.001, all compared models detect more non-errors and lead to more errors than before. But it is obvious that ours take less errors in. On bigger ratios, both CAC and DAC reduce the number of errors while almost all baselines still not. Our proposed methods can be helpful in real-case applications.

	0.001	**0.005**	**0.01**	**0.02**
MDMM	-5.677	-0.874	-0.280	0.010
DAM	-27.543	-4.910	-2.153	-0.780
CSNG	-9.241	-1.399	-0.452	-0.002
CAC	-1.35	0.203	0.349	0.376
DAC	**-1.10**	**0.265**	**0.381**	**0.394**

Table 3. Error Difference Performance Comparison

4.4 Confidence Threshold Analysis

To understand the pattern of our models' confidence, we have an ablation study of confidence threshold parameter t. Fig. 2 illustrates the F1 performance of CAC and DAC

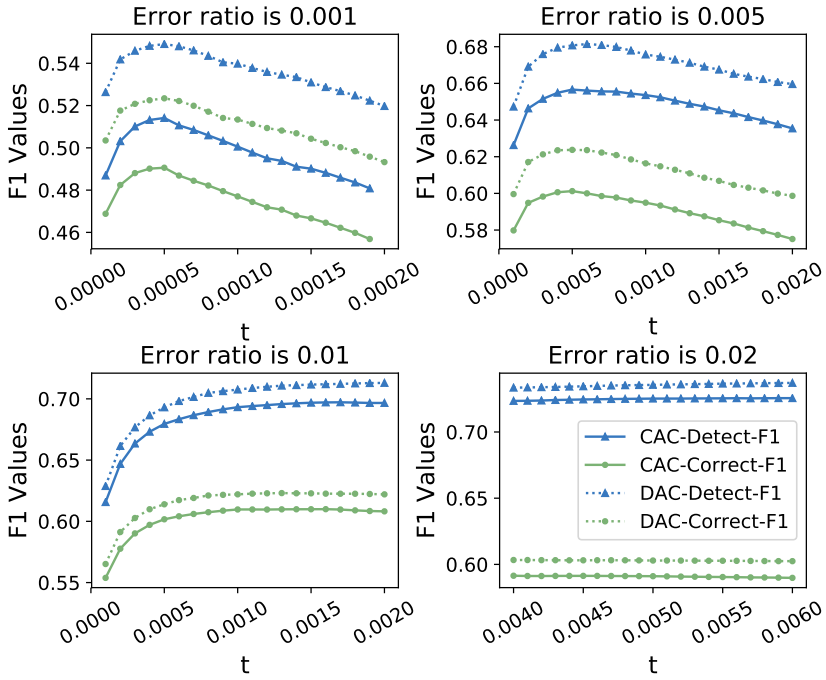


Fig. 2. F1 Against Different Threshold t

against different numbers of t on test sets. We can observe that the trend of detection F1 and correction F1 are almost the same and we can modify the threshold to effectively meet the need of different estimated error ratios. For data of 0.1%, the empirical optimal confidence threshold is 0.00005 and for data of 0.5% it is 0.0006; for data of 1%, it is good to apply 0.0018 and for data of 2%, performances around 0.005 are similar. A lower confidence threshold means more errors are not going to be detected and corrected; on the other hand, a larger threshold gives our model more confidence to detect and correct a spelling error. With the t growing larger than some value, the F1 begins to decrease, which indicates that we can decide an appropriate threshold for our models to gain better performances. By doing such we can also avoid the bad error difference on data of 0.001.

5 Conclusion

In this paper, we proposed two attention based deep models combined with confusion sets used for proofreading spelling errors in character-level. We regarded spelling check as a prediction task, and through attention mechanism we computed the relation score between every candidate characters and the context encoded by LSTMs. For the lack of supervised dataset we define a new methodology for obtaining (preceding text, following text, candidates, target) quads. We evaluate our methods on test sets of four different error ratios and report the state-of-the-art performance.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
2. Elman, J.L.: Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3):195–225, 1991.
3. Liu, C., Lai, M., Chuang, Y., Lee, C.: Visually and phonologically similar characters in incorrect simplified chinese words. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 739–747. Association for Computational Linguistics, 2010.
4. Liu L., Cao, C.: Chinese real-word error automatic proofreading ased on combining of local context features. *Computer Science*, 43(12):30–35, 2016.
5. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.
6. Mays, E., Damerau, F.J., Mercer, R.L.: Context based spelling correction. *Information Processing & Management*, 27(5):517–522, 1991.
7. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.
8. Kinga, D., Adam, J.: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
9. Hermann, K., Grefenstette,E., Espeholt, L., Will Kay, et al.: Teaching machines to read and comprehend. *arxiv.org/abs/1506.03340*, 2015.
10. Tang, D., Qin, B., Feng, X., Ting L.: Effective lstms for target-dependent sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3298–3307, 2016.
11. Tseng, Y.H., Lee, L.H, Chang, L.P., Chen, H.H.: Introduction to sighan 2015 bake-off for chinese spelling check. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*, pages 32–37, 2015.
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones L., et al.: Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.
13. Hearn, A.W., Hirst, G., Budanitsky, A.: Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 605–616. Springer, 2008.
14. Zhang, L., Huang, C., Zhou, M., Pan, H.: Automatic detecting/correcting errors in chinese text by an approximate word-matching algorithm. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 248–254. Association for Computational Linguistics, 2000.
15. Golding, A.R., Roth, D.: A winnow-based approach to context-sensitive spelling correction. *Machine learning*, 34(1-3):107–130, 1999.
16. Hirst, G., Budanitsky, A.: Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, 2005.
17. Zhao, H., Cai, D., Xin, Y., Wang, Y., Jia, Z.: A hybrid model for chinese spelling check. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):21, 2017.
18. Zhuang, L., Bao, T., Zhu, X., Wang, C., Naoi, S.: A chinese ocr spelling check approach based on statistical language models. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 5, pages 4727–4732. IEEE, 2004.