Context-aware Stand-alone Neural Spelling Correction

Xiangci Li [*]
University of Texas at Dallas
Richardson, TX
lixiangci8@gmail.com

Hairong Liu
Baidu USA
Sunnyvale, CA
liuhairong@baidu.com

Liang Huang
Baidu USA
Sunnyvale, CA
lianghuang@baidu.com

Abstract

Existing natural language processing systems are vulnerable to noisy inputs resulting from misspellings. On the contrary, humans can easily infer the corresponding correct words from their misspellings and surrounding context. Inspired by this, we address the *stand-alone* spelling correction problem, which only corrects the spelling of each token without additional token insertion or deletion, by utilizing both spelling information and global context representations. We present a simple yet powerful solution that jointly detects and corrects misspellings as a sequence labeling task by fine-turning a pre-trained language model. Our solution outperform the previous state-of-the-art result by 12.8% absolute $F_{0.5}$ score.

1 Introduction

A spelling corrector is an important and ubiquitous pre-processing tool in a wide range of applications, such as word processors, search engines and machine translation systems. Having a surprisingly robust language processing system to denoise the scrambled spellings, humans can relatively easily solve spelling correction (Rawlinson, 1976). However, spelling correction is a challenging task for a machine, because words can be misspelled in various ways, and a machine has difficulties in fully utilizing the contextual information.

Misspellings can be categorized into *non-word*, which is out-of-vocabulary, and the opposite, *real-word* misspellings (Klabunde, 2002). The dictionary look-up method can detect non-word misspellings, while real-word spelling errors are harder to detect, since these misspellings are in the vocabulary (Mays et al., 1991; Wilcox-O'Hearn et al., 2008). In this work, we address the *stand-alone* (Li et al., 2018) spelling correction problem. It only corrects the spelling of each token without introducing new tokens or deleting tokens, so that the original information is maximally preserved for the down-stream tasks.

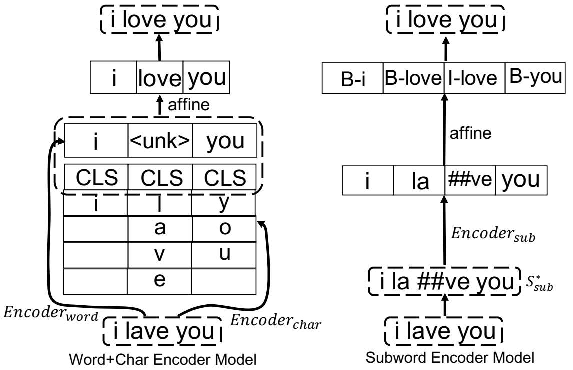

Figure 1: A schematic illustration of our approach. Left: combined word-level and character-level encoder model. Right: subword-level model using $BIO2$ tagging scheme (Sang and Veenstra, 1999).

We formulate the *stand-alone* spelling correction as a sequence labeling task and jointly detect and correct misspellings. Inspired by the human language processing system, we propose a novel solution on the following aspects: (1) We encode both spelling information and global context information in the neural network. (2) We enhance the real-word correction performance by initializing the model from a pre-trained language model (LM). (3) We strengthen the model's robustness on unseen non-word misspellings by augmenting the training dataset with a synthetic character-level noise. As a result, our best model [1] outperforms the previous state-of-the-art result (Wang et al., 2019) by 12.8% absolute $F_{0.5}$ score.

[*] Work performed during internship with Baidu USA

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 407–414
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

2 Approach

We use the transformer-encoder (Vaswani et al., 2017) to encode the input sequences and denote it as $Encoder$. As illustrated in Figure 1, we present both Word+Char encoder and Subword encoder, because we believe the former is better in encoding spelling information, while the latter has the benefit of utilizing a large pre-trained LM.

Word+Char encoder. We use a word encoder to extract global context information and a character encoder to encode spelling information. As shown in equation 1, in order to denoise the noisy word sequence S^* to the clean sequence S, we first separately encode S^* using a word-level transformer-encoder $Encoder_{word}$ and each noisy spelling sequence C_k^* of token k via a character-level transformer-encoder $Encoder_{char}$. For $Encoder_{word}$, we replace non-word misspellings, i.e. OOV words, with a $\langle unk \rangle$ token. For $Encoder_{char}$, we treat each character as a token and each word as a "sentence", so each word's character sequence embedding h_{char}^k is independent of each other. Since the transformer-encoder (Vaswani et al., 2017) computes contextualized token representations, we take h_{char}, the $[CLS]$ token representation of each character sequence as the local character-level representation of S^*. Finally, we jointly predict S by concatenating the local and global context representations.

$$\begin{aligned} h_{word} &= Encoder_{word}(S^*) \\ h_{char}^k &= Encoder_{char}(C_k^*) \\ h_{char} &= [CLS(h_{char}^1), CLS(h_{char}^2), ..., CLS(h_{char}^n)] \\ h_S &= [h_{word}; h_{char}] \\ p(S) &= softmax(Wh_S + b)) \end{aligned} \tag{1}$$

Subword encoder. Alternatively, we use subword tokenization to simultaneously address the spelling and context information. Formally, as shown in equation 2, given a noisy subword token sequence S_{sub}^*, we encode it using a transformer-encoder $Encoder_{sub}$ and simply use an affine layer to predict the sequence of each subword token's corresponding correct word token S_{sub} in $BIO2$ tagging scheme (Sang and Veenstra, 1999).

$$\begin{aligned} h_{sub} &= Encoder_{sub}(S_{sub}^*) \\ p(S_{sub}) &= softmax(W_{sub}h_{sub} + b_{sub}) \end{aligned} \tag{2}$$

Furthermore, we fine-tune our Subword encoder model with a pre-trained LM initialization to enhance the real-word misspelling correction performance.

We use cross-entropy loss as our training objective. Finally, in addition to the natural misspelling noise, we apply a synthetic character-level noise to the training set to enhance the model's robustness to unseen misspelling patterns. The details will be introduced in section 3.1.

3 Experiments

3.1 Dataset

Since we cannot find a sentence-level misspelling dataset, we create one by using the sentences in the 1-Billion-Word-Language-Model-Benchmark (Chelba et al., 2013) as gold sentences and randomly replacing words with misspellings from a word-level natural misspelling list (Mitton, 1985; Belinkov and Bisk, 2017) to generate noisy input sentences. In a real scenario, there will always be unseen misspellings after the model deployment, regardless of the size of the misspelling list used for training. Therefore, we only use 80% of our full word-level misspelling list for *train* and *dev* set. In order to strengthen the robustness of the model to various noisy spellings, we also add noise from a character-level synthetic misspelling list (Belinkov and Bisk, 2017) to the training set. As a result, real-word misspelling contributes to approximately 28% of the total misspellings for both *dev* and *test* set. The details are described in Section A.1

3.2 Results

Performance Metrics We compare word-level precision, recall and $F_{0.5}$ score, which emphasizes precision more. We also provide accuracy for reference in Table 1, because both of the baselines were evaluated with accuracy score. Table 3 shows the definition of true positive (TP), false positive (FP), false negative (FN) and true negative (TN) in this work to avoid confusions. We calculate them using the following equations:

$$\begin{aligned} accuracy &= (TP + TN)/(TP + FP + FN + TN) \\ precision &= TP/(TP + FP) \\ recall &= TP/(TP + FN) \\ F_\beta &= (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \end{aligned}$$

where $\beta = 0.5$ in this work.

[1] https://github.com/jacklxc/StandAloneSpellingCorrection

	Models	Dev				Test			
		Acc	P	R	$F_{0.5}$	Acc	P	R	$F_{0.5}$
1	ScRNN (Sakaguchi et al., 2017)	0.958	0.823	0.890	0.836	0.946	0.755	0.865	0.775
2	MUDE (Wang et al., 2019)	0.966	0.829	0.952	0.851	0.952	0.751	0.928	0.781
3	Char Encoder	0.883	0.517	0.819	0.559	0.870	0.458	0.802	0.501
4	Word Encoder	0.932	0.565	0.949	0.615	0.924	0.521	0.903	0.570
5	Word + Char Encoder	0.988	**0.959**	0.959	**0.959**	0.974	0.882	0.929	0.891
6	+ random char	0.986	0.953	0.947	0.951	0.976	**0.898**	0.927	0.904
7	Subword Encoder	0.986	0.934	0.972	0.941	0.968	0.831	0.950	0.852
8	+ Char Encoder	0.980	0.908	0.959	0.917	0.963	0.808	0.939	0.831
9	+ random char	0.985	0.931	0.966	0.938	0.973	0.866	0.950	0.881
10	+ LM pre-train	**0.990**	0.951	**0.982**	0.957	0.975	0.866	0.962	0.883
11	+ LM pre-train + random char	0.989	0.946	0.979	0.952	**0.980**	0.896	**0.964**	**0.909**

Table 1: Model performance and ablation studies measured by accuracy, precision, recall and $F_{0.5}$.

	Models	Real-Word						Non-Word	
		dev			test			dev	test
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	P
1	ScRNN (Sakaguchi et al., 2017)	0.507	0.592	0.522	0.456	0.523	0.468	0.952	0.873
2	MUDE (Wang et al., 2019)	0.595	0.825	0.630	0.533	0.747	0.566	0.945	0.855
3	Char Encoder	0.106	0.304	0.122	0.099	0.296	0.113	0.886	0.792
4	Word Encoder	**0.916**	0.889	**0.911**	**0.835**	0.792	**0.826**	0.438	0.414
5	Word + Char Encoder	0.900	0.851	0.900	0.819	0.750	0.804	0.979	0.903
6	+ random char	0.902	0.807	0.881	0.819	0.741	0.802	0.969	0.924
7	Subword Encoder	0.804	0.897	0.821	0.715	0.827	0.735	**0.988**	0.877
8	+ Char Encoder	0.740	0.848	0.759	0.664	0.786	0.685	0.978	0.867
9	+ random char	0.799	0.876	0.813	0.718	0.819	0.736	0.984	0.925
10	+ LM pre-train	0.850	**0.935**	0.866	0.771	0.870	0.789	**0.988**	0.877
11	+ LM pre-train + random char	0.845	0.922	0.860	0.787	**0.872**	0.803	0.987	**0.941**

Table 2: Real-word and non-word performance measured by precision, recall and $F_{0.5}$. **All of the recall of non-word is 1.000.**

= Ground Truth?	Noisy Input	Prediction
True Positive	✗	✓
False Positive	✓	✗
False Negative	✗	✗
True Negative	✓	✓

Table 3: Definition of True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). ✓means the noisy input token or prediction the same as the ground truth, and vice versa for ✗.

Baselines. Sakaguchi et al. (2017) proposed semi-character recurrent neural network (ScRNN), which takes the first and the last character as well as the bag-of-word of the rest of the characters as features for each word. Then they used an LSTM (Hochreiter and Schmidhuber, 1997) to predict each original word. Wang et al. (2019) proposed MUDE, which uses a transformer-encoder (Vaswani et al., 2017) to encode character sequences as word representations and used an LSTM (Hochreiter and Schmidhuber, 1997) for the correction of each word. They also used a Gated Recurrent Units (GRU) (Cho et al., 2014) to perform the character-level correction as an auxiliary task during training. We train ScRNN (Sakaguchi et al., 2017) and MUDE (Wang et al., 2019), both of which are *stand-alone* neural spelling correctors, on our dataset as baselines.

Overview. As row 11 of Table 1 shows, fine-tuning the Subword (WordPiece (Peters et al., 2018)) encoder model with LM initialization

(ERNIE 2.0 (Sun et al., 2019)) on the augmented dataset with synthetic character-level misspellings yields the best performance. Without leveraging a pre-trained LM, the Word+Char Encoder model trained on the augmented dataset with synthetic character-level misspellings performs the best (row 6). In fact, the differences between these approaches are small.

In Table 2, we calculate real-word and non-word correction performance to explain the effect of each training technique applied. Note that as shown in Figure 1, because non-word misspellings are preprocessed already, the detection of these non-word misspellings can be trivially accomplished, which results in all models having non-word recall of 1.000.

As Table 2 shows, strong models overall perform well on both real-word misspellings and non-word misspellings. Although our models perform better on non-word misspellings than real-word misspellings, the significant improvement of our models over the baselines comes from the real-word misspellings, due to the usage of the pre-trained LM. In the following paragraphs, we state our claims and support them with our experimental results.

Spelling correction requires both spelling and context information. As Table 2 shows, without the context information, the character encoder model (row 3) performs poorly on real-word misspellings. On the contrary, word encoder model (row 4) performs well on real-word misspellings, but poorly on non-word misspellings, due to the lack of the spelling information. The combined Word+Char encoder model (row 5) leverages both spelling and context information and thus improves nearly 40% absolute $F_{0.5}$ in Table 1. It even outperforms the LM intialized model (row 10). Both of the baseline models (row 1 and 2) perform poorly, because they perform spelling corrections upon character sequences, which disregards the semantics of the context, as their poor real-word performance in Table 2 row 1 and 2 suggests. On the other hand, since subword embeddings essentially subsume character embedding, an additional character encoder does not improve the performance of the Subword encoder model (Table 1 row 8).

Pre-trained LM facilitates spelling correction. As row 10 of Table 1 shows, fine-tuning the model with a pre-trained LM weight initialization improves both precision and recall score over the Subword encoder model (row 7). The LM pre-training mainly improves real-word recall as Table 2 row 10 suggests. Pre-trained LMs are trained with multiple unsupervised pre-training tasks on a much larger corpus than ours, which virtually expands the training task and the training set.

Because most neural language models are trained on the subword level, we are not able to obtain a pre-trained LM initialized version of Word+Char encoder model (row 5). Nonetheless, we hypothesize that such a model will yield a very promising performance given sufficient training data and proper LM pre-training tasks.

Training on additional synthetic character-level noise improves model robustness. As row 6, 9 and 11 of Table 1 and 2 shows, in addition to frequently occurring natural misspellings, training models on the texts with synthetic character-level noise improves the test performance, which is mainly contributed by the improvement of precision on non-word misspellings. Note that the *train* and *dev* set only cover 80% of the candidate natural misspellings. Adding character-level noise in the training data essentially increases the variety of the missplelling patterns, which makes the model more robust to unseen misspelling patterns.

4 Related Work and Background

Many approaches are proposed for spelling correction (Formiga and Fonollosa, 2012; Kukich, 1992; Whitelaw et al., 2009; Zhang et al., 2006; Flor, 2012; Carlson and Fette, 2007; Flor and Futagi, 2012), such as edit-distance based approaches (Damerau, 1964; Levenshtein, 1966; Bard, 2007; Kukich, 1992; Brill and Moore, 2000; De Amorim and Zampieri, 2013; Pande, 2017), approaches based on statistical machine translation (Chiu et al., 2013; Hasan et al., 2015; Liu et al., 2013), and spelling correction for user search queries (Cucerzan and Brill, 2004; Gao et al., 2010). Most of them do not use contextual information, and some use simple contextual features (Whitelaw et al., 2009; Flor, 2012; Carlson and Fette, 2007; Flor and Futagi, 2012).

In recent years, there are some attempts to develop better spelling correction algorithms based on neural nets (Etoori et al., 2018). Similar to our baselines ScRNN (Sakaguchi et al., 2017) and MUDE (Wang et al., 2019), Li et al. (2018) proposed a nested RNN to hierarchically encode characters to word representations, then correct each word using a nested GRU (Cho et al., 2014). However, these previous works either only train models on natural misspellings (Sakaguchi et al., 2017) or synthetic misspellings (Wang et al., 2019), and only focus on denoising the input texts from orthographic perspective without leveraging the retained semantics of the noisy input.

On the other hand, Tal Weiss proposed Deep Spelling (Weiss), which uses the sequence-to-sequence architecture (Sutskever et al., 2014; Bahdanau et al., 2014) to generate corrected sentences. Note that Deep Spelling is essentially not a spelling corrector since spelling correction must focus only on the misspelled words, not on transforming the whole sentences. For similar reasons, spelling correction is also different from GEC (Grammar Error Correction) (Zhang and Wang, 2014; Junczys-Dowmunt et al., 2018).

As a background, recently pre-trained neural LMs (Peters et al., 2018; Devlin et al., 2018; Yang et al., 2019; Radford et al., 2019; Sun et al., 2019) trained on large corpus on various pre-training tasks have made an enormous success on various benchmarks. These LMs captures the probability of a word or a sentence given their context, which plays a crucial role in correcting real-word misspellings. However, all of the LMs mentioned are based on subword embeddings, such as WordPiece (Peters et al., 2018) or Byte Pair Encoding (Gage, 1994) to avoid OOV words.

5 Conclusion

We leverage novel approaches to combine spelling and context information for *stand-alone* spelling correction, and achieved state-of-the-art performance. Our experiments gives insights on how to build a strong *stand-alone* spelling corrector: (1) combine both spelling and context information, (2) leverage a pre-trained LM and (3) use the synthetic character-level noise.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Gregory V Bard. 2007. Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric. In *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*, pages 117–124. Citeseer.

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXiv preprint arXiv:1711.02173*.

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

Andrew Carlson and Ian Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 166–171. IEEE.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S Chang. 2013. Chinese spelling checker based on statistical machine translation. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300.

Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Renato Cordeiro De Amorim and Marcos Zampieri. 2013. Effective spell checking methods using clustering algorithms. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 172–178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. 2018. Automatic spelling correction for resource-scarce languages using deep learning. In *Proceedings of ACL 2018, Student Research Workshop*, pages 146–152.

Michael Flor. 2012. Four types of context for automatic spelling correction. *TAL*, 53(3):61–99.

Michael Flor and Yoko Futagi. 2012. On using context for automatic correction of non-word misspellings in student essays. In *Proceedings of the seventh workshop on building educational applications Using NLP*, pages 105–115. Association for Computational Linguistics.

Lluis Formiga and José AR Fonollosa. 2012. Dealing with input noise in statistical machine translation. In *Proceedings of COLING 2012: Posters*, pages 319–328.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd international conference on computational linguistics*, pages 358–366. Association for Computational Linguistics.

Saša Hasan, Carmen Heger, and Saab Mansour. 2015. Spelling correction of user search queries through statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 451–460.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv preprint arXiv:1804.05940*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ralf Klabunde. 2002. Daniel jurafsky/james h. martin, speech and language processing. *Zeitschrift für Sprachwissenschaft*, 21(1):134–135.

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

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. 2018. Spelling error correction using a nested rnn model and pseudo training data. *arXiv preprint arXiv:1811.00238*.

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.

Eric Mays, Fred J Damerau, and Robert L Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517–522.

Roger Mitton. 1985. Corpora of misspellings for download.

Harshit Pande. 2017. Effective search space reduction for spell correction using character neural embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 170–174.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Graham Ernest Rawlinson. 1976. *The significance of letter position in word recognition*. Ph.D. thesis, University of Nottingham.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 173–179. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhiwei Wang, Hui Liu, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Learning multi-level dependencies for robust word recognition. *arXiv preprint arXiv:1911.09789*.

Tal Weiss. Deep spelling: Rethinking spelling correction in the 21st century.

Casey Whitelaw, Ben Hutchinson, Grace Y Chung, and Gerard Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 890–899. Association for Computational Linguistics.

Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. 2008. 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.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Longkai Zhang and Houfeng Wang. 2014. A unified framework for grammar error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 96–102.

Yang Zhang, Pilian He, Wei Xiang, and Mu Li. 2006. Discriminative reranking for spelling correction. In *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 64–71.

A Appendices

A.1 Dataset Details

We keep the most frequent words in the 1-Billion-Word-Language-Model-Benchmark dataset (Chelba et al., 2013) as our word vocabulary Ψ_w, and all characters in Ψ_w to form our character vocabulary Ψ_c. After deleting sentences containing OOV words, we randomly divide them into three datasets S_{train}, S_{dev} and S_{test}. We merge the two word-level misspelling lists (Mitton, 1985; Belinkov and Bisk, 2017) to get a misspelling list Ω. We randomly choose 80% of all misspellings in Ω to form a *known-misspelling-list*, $\hat{\Omega}$.

To strengthen the robustness of the model to various noisy spellings, we also utilize the methods in Belinkov and Bisk (2017) , namely, *swap*, *middle random*, *fully random* and *keyboard type*, to generate character-level synthetic misspellings. To encourage the model to learn contextual information, we add an additional method, *random generate*, to generate arbitrary character sequences as misspellings.

While replacing gold words with misspellings, for a sentence with n words, the number of replaced words is $m = \max(\lfloor \alpha n \rfloor, 1)$, where $\alpha = \min(|\mathcal{N}(0, 0.2)|, 1.0)$ and $\mathcal{N}$ represents a Gaussian distribution.

The *dev* set is created with misspellings from sampled list $\hat{\Omega}$, and the *test* set is created with misspellings from the full list Ω. We compare 2 *train* sets, the first has only natural misspellings from $\hat{\Omega}$, and the second has natural misspellings as well as synthetic misspellings, which is denoted as $+random\ char$ in Section 3.2. We always use the same *dev* set and *test* set that only contain natural misspellings for comparison.

Table 4 shows the parameters of our *stand-alone* spelling correction dataset. We will release the dataset and codes after this paper is published.

A.2 Implementation Details

We use PaddlePaddle [2] for the network implementation and keep the same configuration for the Subword encoders as ERNIE 2.0 (Sun et al., 2019). We tune the models by grid search on the *dev* set according to $F_{0.5}$ score. The detailed hyper-parameters shown in Table 5. In addition, we use Adam optimizer (Kingma and Ba, 2014) with learning rate of 5e-5 as well as linear decay. We used 10 GeForce GTX 1080 Ti or RTX 2080Ti to train each model until convergence, which takes a few days.

Parameter Name	Value
$\vert\Psi_w\vert$	50000
$\vert\Psi_c\vert$	130
max_sent_len	200
max_word_len	20
$\vert S_1\vert$	17971548
$\vert S_2\vert$	5985
$\vert S_3\vert$	5862

Table 4: Parameters of our *stand-alone* spelling correction dataset.

Parameter Name	Word	Subword	Char
max seq length	256	256	20
hidden size	512	768	256
# hidden layers	6	12	4
# attention heads	8	12	8

Table 5: Hyper-parameters of word encoders, Subword(WordPiece (Wu et al., 2016)) encoders and character encoders.

[2] https://github.com/PaddlePaddle/Paddle