Abstract

Chinese pinyin input methods are very important for Chinese language processing. Actually, users may make typos inevitably when they input pinyin. Moreover, pinyin typo correction has become an increasingly important task with the popularity of smartphones and the mobile Internet. How to exploit the knowledge of users typing behaviors and support the typo correction for acronym pinyin remains a challenging problem. To tackle these challenges, we propose KNPTC, a novel approach based on neural machine translation (NMT). In contrast to previous work, KNPTC is able to integrate explicit knowledge into NMT for pinyin typo correction, and is able to learn to correct a variety of typos without guidance of manually selected constraints or language-specific features. In this approach, we first obtain the transition probabilities between adjacent letters based on large-scale real-life datasets. Then, we construct the “ground-truth” alignments of training sentence pairs by utilizing these probabilities. Furthermore, these alignments are integrated into NMT to capture sensible pinyin typo correction patterns. KNPTC is applied to correct typos in real-life datasets, which achieves 32.77% increment on average in accuracy rate of typo correction compared against the state-of-the-art system.

1 Introduction

Chinese pinyin is the official system to transcribe Chinese characters into the Latin alphabet. Based on this transcription system, pinyin input method has become the dominant method for entering Chinese text into computers in China. The typical way to type in Chinese words is in a sequential manner [Wang et al., 2001]. For example, assuming that users want to type in the Chinese word “你好(Hello)”. First, they type in the corresponding pinyin “nihao” without delimiters such as “Space” key to segment pinyin syllables. Then, a Chinese pinyin input method displays a list of Chinese candidate words which share that pinyin. Finally, users search the target word from candidates and get the result. In this way, typing in Chinese words is more complicated than typing in English words. In fact, pinyin typos have always been a serious problem for Chinese pinyin input methods. Users often make various typos in the process of inputting. As shown in Figure 1, users may mistype “nihao” for “nihap” (the letter p is very close to the letter o on the QWERTY keyboard). In addition, the user may also fail to input the completely right pinyin simply because he/she is a dialect speaker and does not know the exact pronunciation of the expected Chinese word [Jia and Zhao, 2014]. Moreover, with the boom of smartphones, pinyin typos are more prone to be made possibly due to the limited size of soft keyboard, and the lack of physical feedback on the touch screen [Jia and Zhao, 2014]. When an input method engine (IME) fails to correct the typo and produce the desired Chinese sentence, the user have to spend extra effort to move the cursor back to the typo and correct it, which results in a very poor user experience [Jia et al., 2013]. Hence Chinese pinyin typo correction has a significant impact on IME performance.

However, due to the peculiar characteristics of Chinese pinyin, typo correction for Chinese pinyin is quite different from that in other languages. For example, when users want to input “早上好(Good morning)”, they will type “zaoshanghao” instead of segmented pinyin sequence “zao shang hao”. Besides, acronym pinyin input is very com-
mon in Chinese daily input. How to support acronym pinyin input for typo correction is also a challenging problem [Zheng et al., 2011a]. For the above reasons, it is impractical to adopt previous typo correction methods for English to pinyin typo correction directly.

Meanwhile, machine translation is well established in the field of automatic error correction (AEC) [Junczys-Dowmunt and Grundkiewicz, 2016]. With the recent successful work on neural machine translation, neural encoder-decoder models have been used in the AEC as well [Junczys-Dowmunt and Grundkiewicz, 2016; Xie et al., 2016], and have achieved promising results. In this paper, we explore new neural models to tackle the unique challenges of Chinese pinyin typo correction.

The main component of neural machine translation (NMT) systems is a sequence-to-sequence (S2S) model which encodes a sequence of source words into a vector and then generates a sequence of target words from the vector [Li et al., 2017]. Among different variants of NMT, attentional NMT [Bahdanau et al., 2014; Luong et al., 2015] becomes prevalent due to its ability to use the most relevant part of the source statement in each translation step. Unlike the classification-based and statistics-based approaches, NMT is more capable of capturing global contextual information, which is crucial to pinyin typo correction. However, due to the characteristics of Chinese pinyin, to achieve the best performance on pinyin typo correction, we still need to improve the standard NMT model to address several challenges:

- First, due to the limited size of soft keyboard on smartphones, letters close to each other on a Latin keyboard or with similar pronunciations are more likely to be mistyped, which leads to a great proportion of pinyin spelling mistakes [Zheng et al., 2011b]. How to solve these typos caused by the proximity keys is crucial for improving the performance of IMEs.

- Second, since acronym pinyin is widely used in Chinese daily input, typo correction for acronym pinyin input greatly affects the user input experience. However, due to its shortness and sparsity, previous pinyin typo correction methods failed to support the acronym pinyin input.

- Third, in addition to the aforementioned task-specific challenges, the attention quality of NMT is still unsatisfactory. Lack of explicit knowledge may lead to attention faults and generate inaccurate translations [Zhang et al., 2017].

On the one hand, there is a need to introduce prior knowledge into the attention mechanism to improve the performance of NMT. On the other hand, valuable input patterns can be extracted from the user input logs to guide the pinyin typo correction. In order to overcome the aforementioned challenges, in this paper, we propose an approach called "KNPTC", which stands for “Knowledge and Neural machine translation powered Chinese Pinyin Typo Correction”. KNPTC is able to integrate discrete, probabilistic keyboard neighborhoods information about users

pinyin input as prior knowledge into attentional NMT to capture more sensible typo correction patterns. Specifically, we first use large-scale real-life datasets to obtain the transition probabilities between adjacent letters in advance. We then construct the “ground-truth” alignments of training sentence pairs by utilizing these probabilities. Finally, distance between the NMT attentions and the “ground-truth” alignments is computed as a part of the loss function and need to be minimized in the training procedure. Considering that the transition probabilities of adjacent letters provide higher quality knowledge about real-life users' input behaviors, it is expected that the attentional NMT with prior knowledge will achieve better performance on end-to-end pinyin typo correction task.

We find that KNPTC has the following properties: 1) KNPTC can obtain the optimal segmentation and typo correction jointly on the user's original input pinyin sequence; 2) KNPTC is capable of dealing with typos relating to acronym pinyin input. In the experimental study, we compared KNPTC with the Google Input Tool (henceforth referred to as GoogleIT) and the joint graph model proposed in [Jia and Zhao, 2014] (henceforth referred to as JGM). We conduct experiments on real-life datasets and found that KNPTC can achieve significantly better performance than the other two baseline systems.

The contributions of this paper are summarized as follows.

- We propose a solution of exploiting priori knowledge of users’ input behaviors to seamlessly integrate the transition probabilities between adjacent letters with attentional NMT for Chinese pinyin typo correction.

- Compared with previous work, KNPTC is more capable of typo correction for acronym pinyin, which is widely used for Chinese people's daily input.

- We conduct extensive experiments on real-life datasets. The results show that KNPTC significantly outperforms previous systems(such as GoogleIT and JGM) and is efficient enough for practical use.

2 Related Work

Chen and Lee first tried to resolve the problem of Chinese pinyin typo correction [Chen and Lee, 2000]. They designed a statistical typing model to correct the pinyin typos and then used a language model to convert a pinyin sequence to a Chinese characters sequence. However, their method can only model a few rules and thus only a limited number of errors can be corrected. Zheng et al. proposed an error-tolerant pinyin input method called CHIME using a noisy channel model and language-specific features [Zheng et al., 2011a]. Their method assumed that the user inputted pinyin sequence had already been properly segmented by the user, which is inconsistent with the situation in real life. Our model discards this assumption to make it more practical. Jia and Zhao proposed a joint graph model to find the global optimum of pinyin to Chinese conversion and typo

https://www.google.com/inputtools/try/
correction for the input method engine, and achieved better results than previous studies [Jia and Zhao, 2014]. In this paper, we employ this joint graph model as one of our baseline systems.

In addition, Chen et al. attempted to integrate neural network language models into pinyin IME [Chen et al., 2015]. Their main purpose is to improve the decoding predictive performance of IMEs under the premise of ensuring the response speed. However, due to lack of the ability of typo correction, the model they proposed is not convenient to use in real-life situations.

As for automatic error correction in English, most earlier work [Gao et al., 2010; Rozovskaya and Roth, 2010] made use of a lexicon that contains well-spelled words or context features of words. Recently, machine translation is well established in the field of automatic error correction in English [Junczys-Dowmunt and Grundkiewicz, 2016; Xie et al., 2016]. In this work, we mainly focus on the typo correction of Chinese pinyin, which is quite different from other languages for the reasons mentioned before.

3 Our Approach: KNPTC

In this paper, the task of Chinese pinyin typo correction is formulated as a translation task, in which the source language L is the user input pinyin sequence $l_1 l_2 \ldots l_{T_l}$, where $l_i$ is a letter, and the target language S is the correct segmented pinyin syllables $s_1 s_2 \ldots s_{T_s}$, where $s_i$ is a pinyin syllable. For example, taking the user’s misspelled pinyin input sequence “nihapma” as a source sentence, then the corresponding target sentence is “ni hao ma”. KNPTC combines implicit representations of pinyin sequence and explicit knowledge of adjacent letters transitions to further improve the performance of the pinyin typo correction. Figure 2 shows the architecture of KNPTC. It consists of a character-level sequence to word-level sequence model as a backbone, and uses the supervised attention component to jointly learn attention and typo correction.

3.1 NMT Framework as a Backbone for KNPTC

The backbone of KNPTC closely follows the basic neural machine translation architecture with attention proposed in [Bahdanau et al., 2014]. Here we present a key description of this model.

The underlying framework of the model is called RNN EncoderDecoder [Sutskever et al., 2014]. Given a sequence of vectors $x = (x_1, \ldots, x_{T_x})$ representing the source language, the encoder creates corresponding hidden state vectors $e$ using a bidirectional RNN [Schuster and Paliwal, 1997]:

$$e = (h_1, h_{T_x})$$

The hidden state $h_t$ at time $t$ is obtained by concatenating the forward hidden state $\tilde{h}_t$ and the backward one $\tilde{h}_{T_t}$, i.e., $h_t = [\tilde{h}_t; \tilde{h}_{T_t}]^\top$. The $\tilde{h}_t$ and $\tilde{h}_{T_t}$ are computed as follows:

$$\tilde{h}_t = \text{GRU}_{\text{enc}}(\tilde{h}_{t-1}, x_t)$$

$$\tilde{h}_{T_t} = \text{GRU}_{\text{enc}}(\tilde{h}_{T_t+1}, x_t)$$

where $\text{GRU}_{\text{enc}}$ and $\text{GRU}_{\text{enc}}$ represent gated recurrent unit functions. The GRUs with subscripts $\text{enc}$ and $\text{enc}$ indicate the use of two functions with different parameters, respectively. Here, $\text{enc}$ and $\text{enc}$ denote the use of the forward and backward encoder units.

The decoding process also uses the GRU functions, which uses a sequence of hidden states $d_1, d_2, \ldots, d_{T}$ to define the probability of the output sequence $y_1, y_2, \ldots, y_{T_y}$ as follows:

$$p(y_i | y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, d_i, c_i),$$

where $g$ is a nonlinear, multi-layered, function that outputs the probability distribution of $y_i$, and $d_i$ is an RNN hidden state at step $i$, computed as:

$$d_i = \text{GRU}_{\text{dec}}(d_{i-1}, y_{i-1}, c_i).$$

The context vector $c_i$ used to predict $y_i$ is equal to $\sum_{j=1}^{T_y} a_{ij} h_j$. The weight $a_{ij}$ for $h_j$ is given by:

$$a_{ij} = \frac{\exp(q_{ij})}{\sum_{k=1}^{T_x} \exp(q_{ik})},$$

where $q_{ij} = a(d_{i-1}, h_j)$. Here $a$ is a feedforward neural network which is jointly trained with the entire network.

Given a training dataset $\mathcal{C}$ containing $|\mathcal{C}|$ pairs of training samples in the form of $<x, y>$, the training goal of the model is to minimize the cross entropy loss as follows:

$$\text{Loss} = -\sum_{<x, y> \in \mathcal{C}} \sum_{l=1}^{T_y} \log p(y_l | y_{l-1}, x).$$

3.2 Supervised Attention Model with Keyboard Neighborhoods Information

The attention $a_{1,1}, a_{1,2}, \ldots, a_{1,T_y}$ in each step plays an important role in predicting the next word. However, as shown in equation (5), the original attention mechanism, which selectively focuses on the source words, does not take into account any knowledge of the keyboard neighborhoods, which is crucial for Chinese pinyin typo correction. This
knowledge is especially useful when tackling typo correction for acronym pinyin input. Thus, in this section, we introduce discrete and probabilistic keyboard neighborhoods information of users pinyin input to capture the explicit knowledge about users input behaviors for enhancing the attention mechanism and improving pinyin typo correction quality. To integrate the keyboard neighborhoods information into attentional NMT, we propose a novel alignment model using the transition probabilities between adjacent letters. We then use this alignment model to supervise the learning of the attention mechanism.

Assuming that we have such a prior knowledge $p_t$, given a letter $l_i$, we assign a probability $p_t(l_i \rightarrow l_j)$ for any other letter $l_j$. $p_t(l_i \rightarrow l_j)$ represents the probability that the user wants to enter $l_j$ but type in $l_i$ instead since $l_i$ is adjacent to $l_j$ on the keyboard, i.e., $p_t(l_i \rightarrow l_j)$ is the probability that letter $l_j$ transfers to the letter $l_i$. Given a vocabulary $V$, the keys of which are adjacent letters, we define the following distance function between two sequences:

$$d(s, t) = \sum_{i=1}^{\min(|s|,|t|)} \mathbb{1}(s_i \neq t_i)$$

where $\mathbb{1}()$ is the indicator function with $\mathbb{1}(x) = 1$ if $x$ is true, and $\mathbb{1}(x) = 0$ otherwise.

Using Alignment Model to Supervise Attentional NMT
Based on the “ground-truth” alignment matrix $\phi^*$ mentioned above and the attentions $\phi'$ generated by the NMT model, we define the following distance function between $\phi^*$ and $\phi'$:

$$d_{\phi^*}(\phi') = \frac{1}{2} \left( \|\phi^* - \phi'\|_2^2 \right)$$

Figure 3: The original alignment matrix $\phi$ and the alignment matrix $\phi^*$ after transformation.

Compared with the previous function in equation (6), we finally obtain the following loss function:

$$\text{Loss} = - \sum_{<x,y> \in \mathcal{L}} \left\{ \frac{T_x}{\sum_{i=1}^{T_x} \log p(y_i | y_{<i}, x) - \ell(\phi^*, \phi') } \right\}.$$
and the second part represents the loss term about alignment model which incorporates the transition probabilities between adjacent letters. We then minimize them jointly in our training process.

**Obtaining Prior Knowledge \( p_t \) from Data**

In the previous sections, we propose a method of using prior knowledge \( p_t \) to supervise the learning of attention mechanism. Now, we define the ways to obtain the prior knowledge \( p_t \) from the real-life input behaviors of anonymous users.

\[
p_t(l_i \rightarrow l_j) = \frac{N(l_i \rightarrow l_j)}{N(l_i)}, \quad (12)
\]

where \( N(l_i \rightarrow l_j) \) denotes the number of times that the letter \( l_j \) wrongly entered as the letter \( l_j \) in the actual editing actions, and \( N(l_i) \) indicates the total number of times the letter \( l_i \) being inputted.

### 4 Experiments

#### 4.1 Data Preparation

Our real-life datasets come from user logs of the Chinese input method called Sogou-Pinyin.\(^4\) The user logs mainly include the following parts:

- The user’s input string, which is an unsegmented sequence of Latin letters (henceforth referred to as “original input”);
- The desired Chinese sentence selected by the user (henceforth referred to as “target Chinese sentence”) from the candidates, which are generated by the IME and consist of one or more Chinese words;
- The backspace and re-enter operations made by the user will be recorded if the candidates do not meet the user’s needs due to typos or other reasons.

We extracted original input and target Chinese sentence pairs from these data for experiments. Under the normal circumstances, users should input the complete pinyin sequence corresponding to the target Chinese sentence. However, original input may contain abnormal input due to typos. We classified abnormal input into two types: acronym pinyin input and misspelled pinyin input (acronym pinyin input also contains typos). According to the acronym pinyin’s position in the original input, we simply divided acronym pinyin input into two categories:

- **Global acronym pinyin input.** Each of the Chinese characters uses acronym pinyin for representing, for example: “你好→ni hao ma→ni h m”.

- **Local acronym pinyin input.** Only a part of Chinese characters use acronym pinyin for representing, for example: “你好→ ni hao ma→ ni hao ma” or “你好→ ni hao ma→ ni h m”.

| User input      | Target Chinese sentence (in pinyin form) | Input type |
|-----------------|----------------------------------------|------------|
| dadianhuageini  | da dian hua gei ni                     | CP         |
| xianzqub        | xian zai qu ba                         | LAP        |
| yijianzq        | yi jian zhong qing                     | LAP        |
| bgnll           | bu gen ni liao le                      | GAP        |
| niyiubudhi      | ni you bu shi                         | MP         |

Table 1: Some sample data extracted from user logs and its corresponding input type. For convenience, we use CP to indicate the correct pinyin input, LAP to be the local acronym pinyin input, GAP for the global acronym pinyin input and MP for the misspelled pinyin input.

| Input type | #Records | Proportion (%) |
|------------|----------|----------------|
| CP         | 488,583  | 41.58          |
| LAP        | 408,228  | 34.74          |
| GAP        | 211,261  | 17.98          |
| MP         | 66,717   | 5.70           |

Table 2: Input types distribution. We can observe that there is a large proportion of acronym pinyin input in the real-life data.

Table\(^{[1]}\) shows examples that we extracted from the user logs. The distribution of the input types is given in Table\(^{[2]}\). It should be noted that the original input is a continuous sequence without segmentation. In addition, unlike English and other languages, the original input only expresses the pronunciation information of the target Chinese sentence, so we translated the target Chinese sentence into its pinyin form to obtain the target pinyin sentence. We treat the original input as the source sentence and the target pinyin sentence as the target sentence in our approach. Finally, 1,179,789 anonymous users’ typing records during 2 days were collected in total. From the extracted dataset with about 1M parallel sentences, we randomly selected close to 50K parallel sentences for using as a development set, 100K for testing and 850K for training.

#### 4.2 Evaluation Metrics and Baselines

We evaluate KNPTC with conventional sequence labeling evaluation metrics: word accuracy and sentence accuracy, which have significant impacts on user experience of IMEs [Jia and Zhao, 2014].\(^5\)

We compare KNPTC with two approaches: GoogleIT and JGM (state-of-the-art method). GoogleIT contains a practical pinyin input method. We use its public APIs for testing. JGM is proposed in [Jia and Zhao, 2014].\(^5\) The basic idea of JGM is to adopt the graph model for pinyin typo correction and segmentation jointly. We obtain its source code from the author\(^5\) and use its default settings for hyper param-

\(^{4}\)https://pinyin.sogou.com

\(^{5}\)http://bcmi.sjtu.edu.cn/~zhaohai
Table 3: Example typo corrections made by JGM, GoogleIT and KNPTC. Typo correction made by KNPTC is in **bold**.

| Original input | Target sentence | JGM          | GoogleIT     | KNPTC         |
|----------------|-----------------|--------------|--------------|---------------|
| yi jian        | yi jian         | yi jian      | yi jia men   | yi jian       |
| yi ge xiao     | yi ge xiao a    | yi ge ci ao  | yi ge xiao   |
| zhong qi kua le| zhong qi kua a   | zhong qi yi  |
| shou mi shi   | shuo mi shu he  |
| găng shuí      | găng shuí a     | găng shuí a  |
| xíng yì        | găng shuí xíng  |

The examples of typo corrections in Table 3 demonstrate the abilities of KNPTC, which is able to handle a variety of pinyin typos, e.g., due to keyboard neighborhood (qíy → qiú, gāng → gān) or acronym pinyin (smshuh → shen me shi hou). Considering that the users' real-life input contains different types, to further inspect the robustness of KNPTC, we evaluate the performance of each method separately on all input types, i.e., CP, LAP, GAP, MP and mixed input type. Table 4 shows the performance of JGM, GoogleIT and KNPTC on the test dataset. Some observations can be derived: 1) KNPTC performs significantly better than JGM in almost all the cases, especially for LAP and GAP (with 38.05% and 45.93% improvements for W-Acc). From Table 3 we notice that JGM fails to generate the correct sentence for acronym pinyin input due to its shortness and sparsity, while KNPTC performs well on both LAP and GAP. The reason is that KNPTC is more capable of learning the sensible semantic information and typing patterns of user input. Besides, KNPTC is able to capture the alignments information with transition probabilities between adjacent letters in the decoding stage, and thus it can select more relevant input to predict the next target word. It is worth noting that acronym pinyin input occupies a large part in real-life user input. Therefore, KNPTC is more practical than JGM. 2) KNPTC outperforms GoogleIT, especially for MP (with 44.29% and 77.29% improvements for W-Acc and S-Acc). From Table 3 we can observe that GoogleIT fails to correctly segment the original pinyin input due to the existence of typos and therefore it can not produce correct results. KNPTC overcomes this problem by finding the optimal segmentation and typo correction jointly on the user's original input pinyin sequence. We can also observe that KNPTC's S-Acc on GAP is slightly lower than that of GoogleIT. This can be attributed to the fact that the language model of Chinese sentences is not been fully exploited in KNPTC, which is useful for candidates generation. To make typo correction better, we plan to integrate it with KNPTC in future work. 3) KNPTC's S-Acc on LAP, MP and MIX is significantly better than that of GoogleIT and JGM. This means that KNPTC can generate better candidate sentences, which is critical to the user experience of IMEs. From the results in Table 4 we can further conclude that the alignments information with transition probabilities between adjacent letters can help improve the overall performance.

### 5 Conclusion

In this paper, we propose an effective approach called KNPTC to integrate the transition probabilities between adjacent letters into attentional NMT to capture more sensi-
ble typo correction patterns. KNPTC finds the optimal segmentation and typo correction jointly on the user’s original input pinyin sequence. In addition, KNPTC can also cope with the typo correction of acronym pinyin input. Experiments show that KNPTC can evidently improve the pinyin typo correction performance. To our best knowledge, our work is among the earliest studies in leveraging neural machine translation for Chinese pinyin typo correction.

References

[Bahdanau et al., 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014.

[Chen and Lee, 2000] Zheng Chen and Kai-Fu Lee. A new statistical approach to chinese pinyin input. In 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000, 2000.

[Chen et al., 2015] Shenyuan Chen, Hai Zhao, and Rui Wang. Neural network language model for chinese pinyin input method engine. In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, PACLIC 29, Shanghai, China, October 30 - November 1, 2015, 2015.

[Gao et al., 2010] Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. A large scale ranker-based system for search query spelling correction. In Proceedings of the 23rd International Conference on Computational Linguistics, pages 358–366, 2010.

[Ji et al., 2017] Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 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, 2017.

[Jia and Zhao, 2014] Zhongye Jia and Hai Zhao. A joint graph model for pinyin-to-chinese conversion with typo correction. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers, pages 1512–1523, 2014.

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

[Junczys-Dowmunt and Grundkiewicz, 2016] Marcin Junczys-Dowmunt and Roman Grundkiewicz. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, pages 1546–1556, 2016.

[Luong et al., 2015] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, pages 1412–1421, 2015.

[Rozovskaya and Roth, 2010] Alla Rozovskaya and Dan Roth. Generating confusion sets for context-sensitive error correction. In Proceedings of the 2010 conference on empirical methods in natural language processing, pages 961–970, 2010.

[Schuster and Paliwal, 1997] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. IEEE Trans. Signal Processing, 45(11):2673–2681, 1997.

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

[Wagner and Fischer, 1974] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. J. ACM, 21(1):168–173, 1974.

[Wang et al., 2001] Jingtao Wang, Shumin Zhai, and Hui Su. Chinese input with keyboard and eye-tracking: an anatomical study. In Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems, Seattle, WA, USA, March 31 - April 5, 2001, pages 349–356, 2001.

[Xie et al., 2016] Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. Neural language correction with character-based attention. CoRR, abs/1603.09727, 2016.

[Zhang et al., 2017] Jinchao Zhang, Mingxuan Wang, Qun Liu, and Jie Zhou. Incorporating word reordering knowledge into attention-based neural machine translation. 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 1524–1534, 2017.

[Zheng et al., 2011a] Yabin Zheng, Chen Li, and Maosong Sun. CHIME: an efficient error-tolerant Chinese pinyin input method. In IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, pages 2551–2556, 2011.

[Zheng et al., 2011b] Yabin Zheng, Lixing Xie, Zhiyuan Liu, Maosong Sun, Yang Zhang, and Liyun Ru. Why press backspace? understanding user input behaviors in Chinese pinyin input method. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, pages 485–490, 2011.