Robust Neural Machine Translation with ASR Errors

Haiyang Xue[†], **Yang Feng**[†], **Shuhao Gu**[†], **Wei Chen**[‡]
[†] Key Laboratory of Intelligent Information Processing
Institute of Computing Technology, Chinese Academy of Sciences
[‡] AI Interaction Division, Sogou Inc., Beijing
xuehaiyang,fengyang,gushuhao19b@ict.ac.cn
chenweibj8871@sogou-inc.com

Abstract

In many practical applications, neural machine translation systems have to deal with the input from automatic speech recognition (ASR) systems which may contain a certain number of errors. This leads to two problems which degrade translation performance. One is the discrepancy between the training and testing data and the other is the translation error caused by the input errors may ruin the whole translation. In this paper, we propose a method to handle the two problems so as to generate robust translation to ASR errors. First, we simulate ASR errors in the training data so that the data distribution in the training and test is consistent. Second, we focus on ASR errors on homophone words and words with similar pronunciation and make use of their pronunciation information to help the translation model to recover from the input errors. Experiments on two Chinese-English data sets show that our method is more robust to input errors and can outperform the strong Transformer baseline significantly.

1 Introduction

In recent years, neural machine translation (NMT) has achieved impressive progress and has shown superiority over statistical machine translation (SMT) systems on multiple language pairs (Sennrich et al., 2016). NMT models are usually built under the encoder-decoder architecture where the encoder produces a representation for the source sentence and the decoder generates target translation from this representation word by word (Cho et al., 2014; Sutskever et al., 2014; Gehring et al., 2017; Vaswani et al., 2017). Now NMT systems are widely used in real world and in many cases they receive as input the result of the automatic speech recognition (ASR) system.

Despite the great success, NMT is subject to orthographic and morphological errors which can be comprehended by human (Belinkov and Bisk, 2017). Due to the auto-regression of decoding process, translation errors will be accumulated along with the generated sequence. Once a translation error occurs at the beginning, it will lead to a totally different translation. Although ASR technique is mature enough for commercial applications, there are still recognition errors in their output. These errors from ASR systems will bring about translation errors even totally meaning drift. As the increasing of ASR errors, the translation performance will decline gradually (Le et al., 2017). Moreover, the training data used for NMT training is mainly human-edited sentence pairs in high quality and thus ASR errors in the input are always unseen in the training data. This discrepancy between training and test data will further degrade the translation performance. In this paper, we propose a robust method to address the above two problems introduced by ASR input. Our method not only tries to keep the consistency of the training and test data but to correct the input errors introduced by ASR systems.

We focus on the most widely existent substitution errors in ASR results which can be further distinguished into wrong substitution between words with similar pronunciation and wrong substitution between the words with the same pronunciation (known as homophone words). Table 1 shows Chinese-to-English translation examples of these two kinds of errors. Although only one input word changes in the given three source sentences, their translations are quite different. To keep the consistency between training and testing, we simulate these two types of errors and inject them into the training data randomly. To recover from ASR errors, we integrate the pronunciation information into the translation model to recover the two kinds of errors. For words with similar pronunciation(we name it as Sim-Pron-Words), we first predict the

Proceedings of the 1st Workshop on Automatic Simultaneous Translation, pages 15–23

Gold input	这	份	礼	物	饱	含	一	份	深	情.
	zhè	fèn	lǐ	wù	bǎo	hán	yī	fèn	**shēn**	**qíng**.
ASR-HM	这	份	礼	物	饱	含	一	份	申	请.
	zhè	fèn	lǐ	wù	bǎo	hán	yī	fèn	**shēn**	**qǐng**.
ASR-SP	这	份	礼	物	饱	含	一	份	心	情.
	zhè	fèn	lǐ	wù	bǎo	hán	yī	fèn	**xīn**	**qíng**.
Reference	This gift is full of affection.									
Trans-HM	This gift contains an application.									
Trans-SP	This gift is full of mood.									

Table 1: A Chinese-English translation example with ASR errors. "ASR-HM" gives an input sentence with ASR errors on homophone words and "Trans-HM" shows its translation. "ASR-SP" gives an input sentence with ASR errors on words with similar pronunciation and "Trans-SP" denotes its translation.

true pronunciation and then integrate the predicted pronunciation into the translation model. For homophone words, although the input characters are wrong, the pronunciation is correct and can be used to assistant translation. In this way, we get a two-stepped method for ASR inputted translation. The first step is to get a training data close to the practical input, so that they can have similar distribution. The second step is to smooth ASR errors according to the pronunciation.

We conducted experiments on two Chinese-to-English data sets and added noise to the test data sets at different rates. The results show that our method can achieve significant improvements over the strong Transformer baseline and is more robust to input errors.

2 Background

As our method is based on the self-attention based neural machine translation model (Transformer) (Vaswani et al., 2017), we will first introduce Transformer briefly before introducing our method.

2.1 Encoder and Decoder

Encoder The encoder consists of 6 identical layers. Each layer consists of two sub-layers: self-attention followed by a position-wise fully connected feed-forward layer. It uses residual connections around each of the sub-layers, followed by layer normalization. The output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function carried out by the sub-layer itself. The input sequence $\mathbf{x}$ is fed into these two sub-layers, then we can get the hidden state sequence of the encoder:

$$\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_j)$$

where j denotes the length of the input sentence.

Decoder The decoder shares a similar structure with the encoder, which also consists of 6 layers. Each layer has three sub-layers: self-attention, encoder-decoder attention and a position-wise feed-forward layer. It also employs a residual connection and layer normalization at each sub-layer. The decoder uses masking in its self-attention to prevent a given output position from incorporating information about future output positions during training.

2.2 Attention

The attention mechanism in Transformer is the so-called scaled dot product attention which uses the dot-product of the query and keys to present the relevance of the attention distribution:

$$\mathbf{a} = softmax(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}) \quad (1)$$

where the d_k is the dimensions of the keys. Then the weighted values are summed together to get the final results:

$$\mathbf{t} = \sum(\mathbf{a} \odot \mathbf{V}) \quad (2)$$

Instead of performing a single attention function with a single version of queries, keys and values, multi-head attention mechanism get h different versions of queries, keys and values with different projection functions:

$$\mathbf{Q^i}, \mathbf{K^i}, \mathbf{V^i} = \mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V, i \in [1, h] \quad (3)$$

where $\mathbf{Q}^i, \mathbf{K}^i, \mathbf{V}^i$ are the query , key and value representations of the i-th head respectively. $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are the transformation matrices. h is the number of attention heads. h attention

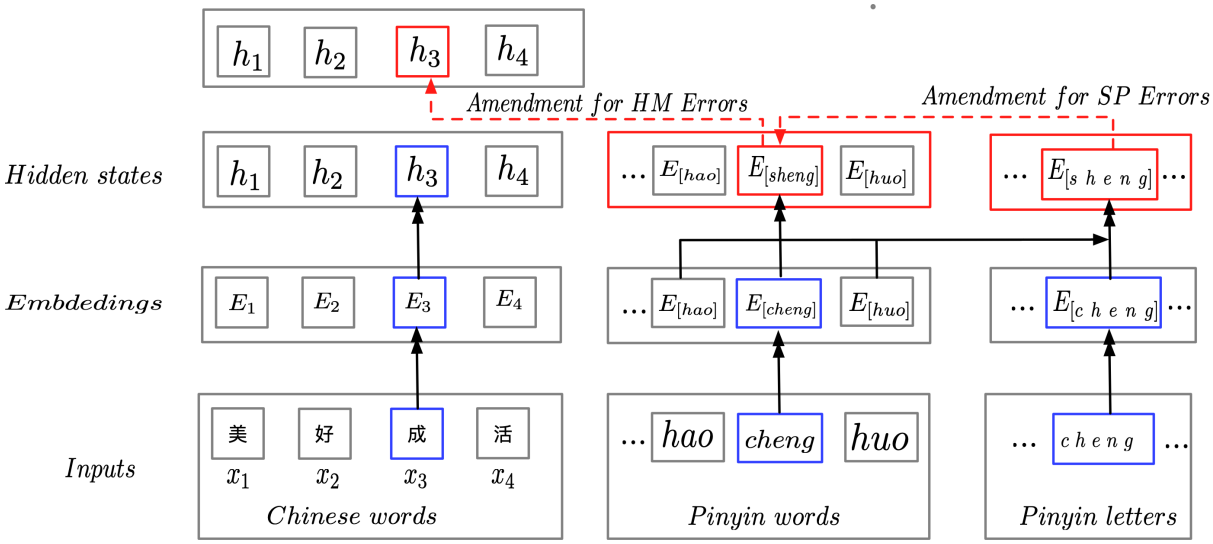

Figure 1: The illustration of our method. "HM" stands for substitution errors between homophone words and "SP" stands for substitution errors between the words with similar pronunciation. The elements in blue boxes are a case of SP errors. Those in the red boxes represent the corrected version with the help of pronunciation information.

Error type	Rate				
Ground Truth	-	语	音	翻	译.
		yǔ	yīn	fān	yì.
Substitution	6.4%	语	音	翻	一.
		yǔ	yīn	fān	yī.
Deletion	2.3%		音	翻	译.
			yīn	fān	yì.
Insertion	0.7%	语	音	翻	了.
		yǔ	yīn	fān	le .

Table 2: Word error rate (WER) against all the words for the three types of ASR errors.

functions are applied in parallel to produce the output states $\mathbf{u}_i$. Finally, the outputs are concatenated to produce the final attention:

$$\mathbf{t} = \text{Concat}(\mathbf{t}_1, ..., \mathbf{t}_h) \quad (4)$$

3 The Proposed Method

Although ASR is mature for commercial applications, there are still recognition errors in the result of ASR. The ASR recognition errors can be classified into three categories: *substitution*, *deletion* and *insertion*, which are shown in Table 2. We counted the word error rate (WER) for the three types of errors respectively on our in-house data set, which consists of 100 hours of Chinese speech across multiple domains. The results in Table 2 gives the ratio of the wrong words against the total words. We can see that the substitution errors are the main errors which is consistent with the results in Mirzaei et al. 2016. Other researchers have proven that over 50% of the machine translation errors are associated with substitution errors which have a greater impact on translation quality than deletion or insertion errors (Vilar et al., 2006; Ruiz and Federico, 2014). Substitution errors can be further divided into two categories: substitution between the words with similar pronunciation (denoted as *SP errors*) and substitution between homophone words (denoted as *HM errors*). Based on these conclusions, we focus on these two kinds of substitution errors in this paper. In what follows we will take Chinese as an example to introduce our method and our method can also be applied to many other languages in a similar way.

Our method aims to improve the robustness of NMT to ASR errors. To this end, our method first constructs a training data set which has a similar data distribution with the test data, then makes use of pronunciation information to recover from the SP errors and HM errors. Specifically, our method works in a flow of three steps as

1. adding SP errors and HM errors in the training data randomly to simulate ASR errors occurring in test;

2. predicting the true pronunciation for SP errors and amending the pronunciation to the predicted results;

3. integrating pronunciation information into the

word semantic to assistant the translation of HM errors as homophone words always have the pronunciation.

Figure 1 illustrate the architecture of our method.

Note that the above three steps must be cascaded which means we always first try to correct the pronunciation information for SP errors and then use the corrected pronunciation information to play a part in the translation for HM errors. We will introduce the three steps in details in the following sections.

3.1 Simulating ASR errors in Training

We process source words one by one by first deciding whether to change it to ASR noise at a certain probability $p \in [0, 1]$, and if yes, then selecting one word to substitute the source word according to the word frequency of the training data. Given a source word x, we first collect its SP word set $\mathcal{V}_{sp}(x)$ and HM word set $\mathcal{V}_{hm}(x)$, then sample from a Bernoulli distribution with a probability p to substitute it with a noise:

$$r_x \sim \text{Bernoulli}(p) \tag{5}$$

where $r_x \in \{0, 1\}$ is the output of the Bernoulli distribution and $p \in [0, 1]$ is the probability that the Bernoulli distribution outputs 1. When r_x is 1, we go to the next step to substitute x. Next, we can select a word to substitute x from a word set $\mathcal{V}(x)$ at a probability as

$$p(x) = \frac{\text{Count}(x)}{\sum\limits_{x' \in \mathcal{V}(x) \setminus \{x\}} \text{Count}(x')} \tag{6}$$

where $\text{Count}(x)$ stands for the count that the word x occurs in the training data, and $\mathcal{V}(x)$ can be $\mathcal{V}_{sp}(x)$, $\mathcal{V}_{hm}(x)$ or $\mathcal{V}_{sp}(x) \cup \mathcal{V}_{hm}(x)$ depending on whether we want to simulate SP errors, HM errors or mixture. To get the training data with the data distribution consistent with the ASR input, we sample words from $\mathcal{V}_{sp}(x) \cup \mathcal{V}_{hm}(x)$.

3.2 Amending Pronunciation for SP Errors

In Chinese, the Pinyin word is used to represent the pronunciation of the word and a Pinyin word usually consists of several Pinyin letters. For example, in Table 2, the Pinyin word for the word "语" is "yǔ" and it has two Pinyin letters as "y" and "ǔ". According to the pronunciation, one Pinyin word can be divided into two parts: *the initial*, which usually only contains the first Pinyin letter, and *the final*, which usually contains the rest Pinyin letters. We looked into our in-house ASR results and found that most SP errors are caused by the wrong initial. Besides, Chinese Pinyin has fixed combinations of the initial and the final, and hence given a final, we can get all possible initials that can occur together with the final in one Pinyin word. In this sense, for an SP error, we can draw the distribution over all the possible initials to predict the correct Pinyin word. With the distribution, we can amend the embedding of the Pinyin word to the correct one.

Formally, given a source sentence $\mathbf{x} = (x_1, \ldots, x_J)$, we use $\mathbf{u} = (u_1, \ldots, u_J)$ to denote its Pinyin word sequence and use u_{jk} to denote the k-th Pinyin letter in the Pinyin word u_j. For a Pinyin word u_j, we represent its initial as

$$u_j^{\text{ini}} = u_{j1} \tag{7}$$

and represent its final as

$$u_j^{\text{fin}} = [u_{j2}, \ldots, u_{jK_j}] \tag{8}$$

where K_j is the number of Pinyin letters of u_j. We also maintain an embedding matrix for the Pinyin words and the Pinyin letters, respectively. Then we can get the embedding for the final u_j^{fin} by adding all the embedding of its Pinyin letters as

$$\mathbf{E}[u_j^{\text{fin}}] = \sum_{k=2}^{K_j} (\mathbf{E}[u_{jk}]) \tag{9}$$

where $\mathbf{E}[\cdot]$ means the corresponding embedding of the input. As SP errors usually result from wrong initials, we predict the probability of the true initial according to the co-occurrence with the immediately previous Pinyin word u_{j-1} and the right after Pinyin word u_{j+1}. Then we can draw the distribution over all the possible initials for u_j as

$$p^{\text{ini}} \sim \text{softmax}(g^{\text{ini}}(\mathbf{E}[u_{j-1}] + \mathbf{E}[u_j^{\text{fin}}] + \mathbf{E}[u_{j+1}])) \tag{10}$$

where $g^{\text{ini}}(\cdot)$ is a linear transformation function. Then we use the weighted sum of the embedding of all the possible initials as the true embedding of u_j^{ini} as

$$\mathbf{E}[u_j^{\text{ini}}] = \sum_{l \in \mathcal{V}^{\text{ini}}(u_j)} p^{\text{ini}}(l) * \mathbf{E}[l] \tag{11}$$

where $\mathcal{V}^{\text{ini}}(u_j)$ denotes the letter set which can be used as the initial of u_j and $p^{\text{ini}}(l)$ denotes the

predicted probability for the Pinyin letter l in Equation 10. Then we can update the embedding of u_j based on the amended Pinyin letter embedding as

$$\mathbf{E}[u_j] = g(\mathbf{E}[u_j], \mathbf{E}[u_j^{\text{ini}}], \mathbf{E}[u_j^{\text{fin}}]) \quad (12)$$

where $g(.)$ is a linear transformation function.

3.3 Amending Encoding for HM Errors

For HM errors, although the source word is not correct, the Pinyin word is still correct. Therefore, the Pinyin word can be used to provide additional true information about the source word. Specifically, we integrate the embedding of Pinyin words into the final output of the encoder, denoted as $\mathbf{h} = (\mathbf{h}_1, \ldots, \mathbf{h}_J)$, to get an advanced encoding for each source word. This is implemented via a gating mechanism and we calculate the gate λ_j for the j-th source word as

$$\lambda_j = \mathbf{W}_\lambda \tanh\left(\mathbf{W}_h \mathbf{h}_j + \mathbf{W}_u \mathbf{E}[u_j]\right) \quad (13)$$

where $\mathbf{W}_\lambda$, $\mathbf{W}_h$ and $\mathbf{W}_u$ are weight matrices. With the gate, we update the hidden state $\mathbf{h}_j$ to

$$\mathbf{h}_j = \lambda_j * \mathbf{h}_j + (1 - \lambda_j) * \mathbf{E}[u_j] \quad (14)$$

Then the updated hidden states of source words are fed to the decoder for the calculation of attention and generation of target words.

4 Experiments

4.1 Data Preparation

We evaluated our method on two Chinese-English data sets which are from the NIST translation task and WMT17 translation task, respectively. For the NIST translation task, the training data consists of about 1.25M sentence pairs from LDC corpora with 27.9M Chinese words and 34.5M English words respectively [1]. We used NIST 02 data set as the development set and NIST 03, 04, 05, 06, 08 sets as the **clean** test sets which don't have ASR errors in the source side. For the WMT17 translation task, the training data consists of 9.3M bilingual sentence pairs obtained by combing the CWMT corpora and News Commentary v12. We use the newsdev2017 and newstest2017 as our development set and clean test set, respectively.

For both of these two corpus, we tokenized and truecased the English sentences using the Moses scripts[2]. Then 30K merging operations were performed to learn byte-pair encoding(BPE) (Sennrich et al., 2015). As for the Chinese data, we split the sentence into Chinese chars. We use the ChineseTone[3] tool to convert Chinese characters into their Pinyin counterpart without tones.

Then we apply the method mentioned in the section 3.1 to add SP errors, HM errors or both to the clean training set to get three kinds of noisy data. We have also set the substituting probability p to 0.1, 0.2 and 0.3 to investigate the impacts of the ASR errors in the training set. Considering that there is no public test sets simulating the substitution errors of ASR, we also crafted another three **noisy** test sets based on the clean sets with different amount of HM errors and SP errors in each source side sentence to test the robustness of the NMT model. We try our best to make these noisy test sets be close to the results of ASR, so that it can check the ability of our proposed method in the realistic speech translation scenario.

4.2 Training Details

We evaluate the proposed method on the Transformer model and implement on the top of an open-source toolkit Fairseq-py (Edunov et al., 2017). We follow (Vaswani et al., 2017) to set the configurations and have reproduced their reported results on the Base model. All the models were trained on a single server with eight NVIDIA TITAN Xp GPUs where each was allocated with a batch size of 4096 tokens. Sentences longer than 100 tokens were removed from the training data. For the base model, we trained it for a total of 100k steps and save a checkpoint at every 1k step intervals. The single model obtained by averaging the last 5 checkpoints were used for measuring the results.

During decoding, we set beam size to 5, and length penalty α=0.6 (Wu et al., 2016). Other training parameters are the same as the default configuration of the Transformer model. We report case-sensitive NIST BLEU (Papineni et al., 2002) scores for all the systems. For evaluation, we first merge output tokens back to their untokenized representation using *detokenizer.pl* and then use *multi-bleu.pl* to compute the scores as per reference.

4.3 Main Results

The main results are shown in the Table 3 and Table 5 (Row1 and Row4). It shows that our propoed

[1]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[2]http://www.statmt.org/moses/

[3]https://github.com/letiantian/ChineseTone

System	p	Clean	Noise			
			1 Sub	2 Subs	3 Subs	Ave.
Baseline	-	45.21	43.63	42.24	41.33	42.40
Our Method	0.1	45.15	44.64	44.23	43.87	44.24
	0.2	45.13	44.83	44.41	44.12	44.45
	0.3	44.95	44.68	44.45	44.09	44.40

Table 3: Case-sensitive BLEU scores of our approaches on thec NIST clean test set (average bleu score on nist03, nist04, nist05, nist06) and three artificial noisy test sets (1 Sub, 2 Subs and 3 Subs) which are crafted by randomly substituting one, two and three original characters of each source sentence in the clean test set with HM errors or SP errors, respectively. p is the substitution rate.

System	p	Clean	Noise Ave.
Baseline	-	45.21	42.40
+SP Amendment	0.2	45.20	43.55
+HM Amendment	0.2	45.30	43.77
+Both Amendment	0.2	45.13	44.45

Table 4: Results of the ablation study on the NIST data. "+SP Amendment", "+HM Amendmen" and "+Both Amendment" represents the model only with the amending pronunciation for SP errors, amending errors for HM errors and with amending pronunciation for both of these two kinds of errors, respectively.

System	p	Clean	Noise
Baseline	-	23.11	20.23
+SP Amendment	0.2	23.08	22.12
+HM Amendment	0.2	23.09	22.23
+Both Amendment	0.2	23.13	22.67

Table 5: Comparison of "+SP Amendment", "+HM Amendmen" and "+Both Amendment" on the WMT17 ZH→EN dataset.

Figure 2: Training cost of the baseline model (blue dots) and our proposed method (red dots).

model significantly outperforms the baseline model on the **noisy** test sets on both of the NIST and WMT17 translation tasks. Furthernmore, we got the following conclusions:

First, the baseline model performs well on the **clean** test set, but it suffers a great performance drop on the **noisy** test sets, which indicates that the conventional NMT is indeed fragile to permuted inputs, which is consistent with prior work (Belinkov and Bisk, 2017; Cheng et al., 2018).

Second, the results of our proposed method show that our model can not only get a competitive performance compared to the baseline model on the clean test set, but also outperform all the baseline models on the noisy tests. Moreover, our proposed method doesn't drop so much on the noisy test sets as the ASR errors increase, which proves that our proposed method is more robust to the noisy inputs after we make use of the pronunciation features to amend the representation of the input tokens for the SP errors and HM errors.

Last, we find that our method works best when the hyper-parameter p was set to 0.2 in our experiments. It indicates that the different noise sampling methods have different impacts on the final results. Too few or too much ASR errors simulated in the training data both can't make the model achieve the best performance in practice. This finding can guide us to better simulate the noisy data, thus helping us train a more robust model in the future work.

4.4 Ablation Study

In order to further understand the impact of the components of the proposed method, we performed some further studies by training multiple versions of our model by removing the some components of it. The first one is just with the amending pronunciation for SP errors. The second one is just with the

amending errors for HM error. The overall results are shown in the Table 4 and Table 5.

The "+SP Amendment" method also improve the robustness and fault tolerance of the model. It is obvious that in all the cases, our proposed Sim-Pron-Words model outperforms baseline system by +1.15 and + 1.89 BLEU. which indicates that it can also greatly enhance the anti-noise capability of the NMT model.

The "+HM Amendmen" method provides further robustness improvements compared to the baseline system on all the noisy test sets. The results show that the model with SP amendment achieves a further improvement by an average of +1.37 and +2.00 BLEU on the NIST and WMT17 noisy test sets respectively. In addition, it has also achieved a performance equivalent to baseline on the clean test sets. It demonstrates that homophones feature is an effective input feature for improving the robustness of Chinese-sourced NMT.

Eventually, as expectecd, the best performance is obtained with the simultaneous use of all the tested elements, proving that these two features can cooperate with each other to improve the performance further.

4.5 Training Cost

We also investigate the training cost of our proposed method and the baseline system. The loss curves are shown in the Figure 2. It shows that the training cost of our model is higher than the baseline system, which indicates that our proposed model may take more words into consideration when predicting the next word, because it aggregate the pronunciation information of the source side character. Thus we can get a higher BLEU score on the test sets than the baseline system, which will ignore some more appropriate word candidates just without the pronunciation information. The training loss curves and the BLEU results on the test sets show that our approach effectively improves the generalization performance of the conventional NMT model trained on the clean training data.

4.6 Effect of Source Sentence Length

We also evaluate the performance of our proposed method and the baseline on the noisy test sets with different source sentence lengths. As shown in Figure 3, the translation quality of both systems is improved as the length increases and then degrades as the length exceeds 50. Our observation is also consistent with prior work (Bahdanau et al., 2014).

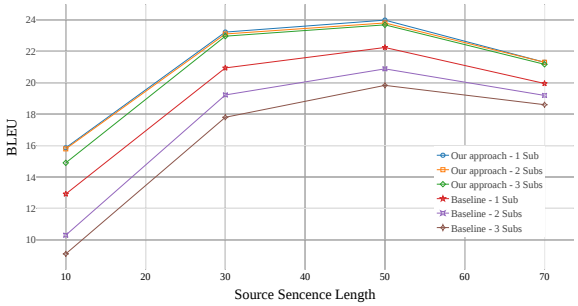

Figure 3: Effect of source sentence lengths of noisy input.

These curves imply that more context is helpful to noise disambiguation. It also can be seen that our robust system outperforms the baseline model on all the noisy test sets in each length interval. Besides, the increasing number of the error in the source sentence doesn't degrade the performance of our proposed model too much, indicating the effectiveness of our method.

4.7 A Case Study

In Table 6, we provide a realistic example to illustrate the advantage of our robust NMT system on erroneous ASR output. For this case, the syntactic structure and meaning of the original sentence are destroyed since the original character "数" which means digit is misrecognized as the character "书" which means book. "数" and "书" share the same pronunciation without tones. Human beings generally have no obstacle to understanding this flawed sentence with the aid of its correct pronunciation. The baseline NMT system can hardly avoid the translation of "书" which is a high-frequency character with explicit word sense. In contrast, our robust NMT system can translate this sentence correctly. We also observe that our system works well even if the original character "数" is substituted with other homophones, such as "舒" which means comfortable. It shows that our system has a powerful ability to recover the minor ASR error. We consider that the robustness improvement is mainly attributed to our proposed ASR-specific noise training and Chinese Pinyin feature.

5 Related Work

It is necessary to enhance the robustness of machine translation since the ASR system carries misrecognized transcriptions over into the downstream MT system in the SLT scenario. Prior work at-

Speech	该	**数**	字	已	经	大	幅	下	降	近	一	半。
	gāi	**shù**	zì	yǐ	jīng	dà	fú	xià	jiàng	jìn	yī	bàn。
ASR	该	书	字	已	经	大	幅	下	降	近	一	半。
	gāi	**shū**	zì	yǐ	jīng	dà	fú	xià	jiàng	jìn	yī	bàn。
Ref	The figure has fallen sharply by almost half.											
Baseline	The book has fallen by nearly half.											
Our Approach	The figure has fallen by nearly half.											

Table 6: For the same erroneous ASR output, translations of the baseline NMT system and our robust NMT system.

tempted to induce noise by considering the realistic ASR outputs as the source corpora used for training MT systems (Peitz et al., 2012; Tsvetkov et al., 2014). Although the problem of error propagation could be alleviated by the promising end-to-end speech translation models (Serdyuk et al., 2018; Bérard et al., 2018). Unfortunately, there are few training data in the form of speech paired with text translations. In contrast, our approach utilizes the large-scale written parallel corpora. Recently, Sperber et al. (2017) adapted the NMT model to noise outputs from ASR, where they introduced artificially corrupted inputs during the training process and only achieved minor improvements on noisy input but harmed the translation quality on clean text. However, our approach not only significantly enhances the robustness of NMT on noisy test sets, but also improves the generalization performance.

In the context of NMT, a similar approach was very recently proposed by Cheng et al. (2018), where they proposed two methods of constructing adversarial samples with minor perturbations to train NMT models more robust by supervising both the encoder and decoder to represent similarly for both the perturbed input sentence and its original counterpart. In contrast, our approach has several advantages: 1) our method of constructing noise examples is efficient yet straightforward without expensive computation of words similarity at training time; 2) our method has only one hyper-parameter without putting too much effort into performance tuning; 3) the training of our approach performs efficiently without pre-training of NMT models and complicated discriminator; 4) our approach achieves a stable performance on noise input with different amount of errors.

Our approach is motivated by the work of NMT incorporated with linguistic input features (Sennrich and Haddow, 2016). Chinese linguistic features, such as radicals and Pinyin, have been demonstrated effective to Chinese-sourced NMT (Liu et al., 2019; Zhang and Matsumoto, 2017; Du and Way, 2017) and Chinese ASR (Chan and Lane, 2016). We also incorporate Pinyin as an additional input feature in the robust NMT model, aiming at improving the robustness of NMT further.

6 Conclusion

Voice input has become popular recently and as a result, machine translation systems have to deal with the input from the results of ASR systems which contains recognition errors. In this paper we aim to improve the robustness of NMT when its input contains ASR errors from two aspects. One is from the perspective of data by adding simulated ASR errors to the training data so that the training data and the test data have a consistent distribution. The other is from the perspective of the model itself. Our method takes measures to handle two types of the most widely existent ASR errors: substitution errors between the words with similar pronunciation (SP errors) and substitution errors between homophone words (HM errors). For SP errors, we make use of the context pronunciation information to correct the embedding of Pinyin words. For HM errors, we use pronunciation information directly to amend the encoding of source words. Experiment results prove the effectiveness of our method and the ablation study indicates that our method can handle both the types of errors well. Experiments also show that our method is stable during training and more robust to the errors.

References

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

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. In *Proc. ICLR*.

Alexandre Bérard, Laurent Besacier, Ali Can Kocabiyikoglu, and Olivier Pietquin. 2018. End-to-end automatic speech translation of audiobooks. In *Proc. ICASSP*.

William Chan and Ian Lane. 2016. On online attention-based speech recognition and joint mandarin character-pinyin training. In *Proc. Interspeech*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards robust neural machine translation. In *Proc. ACL*, pages 1756–1766.

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*.

Jinhua Du and Andy Way. 2017. Pinyin as subword unit for chinese-sourced neural machine translation. In *Irish Conference on Artificial Intelligence and Cognitive Science*.

Sergey Edunov, Myle Ott, and Sam Gross. 2017. https://github.com/pytorch/ fairseq.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Ngoc-Tien Le, Benjamin Lecouteux, and Laurent Besacier. 2017. Disentangling asr and mt errors in speech translation. *arXiv preprint arXiv:1709.00678*.

Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2019. Robust neural machine translation with joint textual and phonetic embedding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049.

Maryam Sadat Mirzaei, Kourosh Meshgi, and Tatsuya Kawahara. 2016. Automatic speech recognition errors as a predictor of l2 listening difficulties. In *Proc. the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, pages 192–201.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. the 40th annual meeting on association for computational linguistics*, pages 311–318.

Stephan Peitz, Simon Wiesler, Markus Nußbaum-Thom, and Hermann Ney. 2012. Spoken language translation using automatically transcribed text in training. In *Proc. IWSLT*.

Nicholas Ruiz and Marcello Federico. 2014. Assessing the impact of speech recognition errors on machine translation quality. *Association for Machine Translation in the Americas (AMTA), Vancouver, Canada*, pages 261–274.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proc. WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proc. the First Conference on Machine Translation*.

Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio. 2018. Towards end-to-end spoken language understanding. *arXiv preprint arXiv:1802.08395*.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. *arXiv preprint arXiv:1704.00559*.

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.

Yulia Tsvetkov, Florian Metze, and Chris Dyer. 2014. Augmenting translation models with simulated acoustic confusions for improved spoken language translation. Proc. ACL.

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 6000–6010.

David Vilar, Jia Xu, D'Haro Luis Fernando, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proc. LREC*, pages 697–702.

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*.

Jinyi Zhang and Tadahiro Matsumoto. 2017. Improving character-level japanese-chinese neural machine translation with radicals as an additional input feature. In *Asian Language Processing (IALP), 2017 International Conference on*, pages 172–175. IEEE.