Addressing the Under-Translation Problem from the Entropy Perspective

Yang Zhao,[1,2*] **Jiajun Zhang,**[1,2] **Chengqing Zong,**[1,2,3] **Zhongjun He,**[4] **and Hua Wu**[4]
[1]National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
[4]Baidu Inc., Beijing, China
{yang.zhao, jjzhang, cqzong}@nlpr.ia.ac.cn, {hezhongjun,wu_hua} @baidu.com

Abstract

Neural Machine Translation (NMT) has drawn much attention due to its promising translation performance in recent years. However, the under-translation problem still remains a big challenge. In this paper, we focus on the under-translation problem and attempt to find out what kinds of source words are more likely to be ignored. Through analysis, we observe that a source word with a large translation entropy is more inclined to be dropped. To address this problem, we propose a coarse-to-fine framework. In coarse-grained phase, we introduce a simple strategy to reduce the entropy of high-entropy words through constructing the pseudo target sentences. In fine-grained phase, we propose three methods, including pre-training method, multitask method and two-pass method, to encourage the neural model to correctly translate these high-entropy words. Experimental results on various translation tasks show that our method can significantly improve the translation quality and substantially reduce the under-translation cases of high-entropy words.

Introduction

Neural machine translation (NMT) based on the encoder-decoder architecture becomes the new state-of-the-art method due to distributed representation and end-to-end learning (Kalchbrenner and Blunsom 2013; Cho et al. 2014; Bahdanau, Cho, and Bengio 2015; Wu et al. 2016; Gehring et al. 2017; Vaswani et al. 2017).

However, NMT still has a drawback that some source words sometimes are mistakenly dropped by the neural model, referring as under-translation problem (Tu et al. 2016; Mi et al. 2016). Fig. 1 shows an example that the source sub-words in red "jia(false)@@" and "e(bad)@@" are missed by the neural model. Actually, the statistics from our analysis and previous studies (Zheng et al. 2018) both show that the current neural models suffer heavily from the under-translation problem even for the state-of-the-art Transformer model (Vaswani et al. 2017).

Several studies focus on the under-translation problem and tend to address this problem by i) improving the attention mechanism, e.g., the coverage model (Tu et al. 2016; Mi et al. 2016), or ii) optimizing the hidden states of the encoder and decoder, e.g., the reconstructing model (Tu et al. 2017) and the modeling past and future method (Zheng et al. 2018). The above methods study the problem in the model level. Different from them, we dive into the word level, and try to answer the following two questions:

source (word): 最终 真善美 彻底 打败 了 假恶丑
source (sub-word): 最终 真@@ 善@@ 美 彻底 打败 了 **假@@ 恶@@ 丑**
pinyin: (sub-word): zuizhong zhen@@ shan@@ mei chedi dabai le **jia@@ e@@ chou**
reference: finally , the true , the good and the beautiful completely defeated **the false**, **the bad** and **the ugly**
NMT: eventually , true , good , and beauty thoroughly defeated **the ugly**

Figure 1: Example to show the under-translation in NMT. The source sub-words (Sennrich, Haddow, and Birch 2016) "jia@@" and "e@@" are missed by the NMT model.

1) Why are some source words missed while the others are not? In the example of Fig. 1, we are curious about the questions that why the sub-words "jia@@" and "e@@" are dropped while the sub-word "chou" is not. Semantically, these three sub-words are coordinate.

2) If we can know that some certain kinds of words are more likely to be missed by the NMT model, how can we reduce the risk of under-translations of these words?

To answer the first question, we analyze the NMT translation results by manual analysis and automatic analysis. Through analysis, we reach the following observation: a source word with a higher **translation entropy** (defined in Section 3) has a larger probability to be dropped by NMT model (both the analysis procedure and statistical results are described in Section 3). We define the translation entropy to measure the translation uncertainty of a source word. The larger the translation entropy of a source word is, the higher translation uncertainty this word is.

To address the under-translation problem of high-entropy words, we propose in this paper a coarse-to-fine framework. We first build pseudo target sentences (coarse-grained

*Contribution during internship at Baidu.

phase) to reduce the entropy of high-entropy words, then the derived pseudo target sentences are utilized to improve the neural model (fine-grained phase).

Specifically, in coarse-grained phase, we construct the pseudo target sentences by replacing the candidate translations of each high-entropy word with its respective special pseudo token. By doing so, the derived pseudo target sentence can sharply reduce the entropy of these words. In fine-grained phase, we propose three methods, including pre-training method, multitask method, and two-pass method, to augment the original model with the derived pseudo target sentences. In the pre-training method, we utilize the derived pseudo target sentences to provide a better parameter initiation. In the multitask method, both the original target sentences and the pseudo target sentences are utilized to train the neural model. In the two-pass method, we divide the translation process into two steps: 1) translation step: to guarantee high-entropy words could be correctly translated to the special pseudo token; 2) disambiguation step: to transform the special pseudo token to the true target token. We test our methods on Chinese-to-English, English-to-Japanese and English-to-German translation tasks. The experimental results demonstrate that the translation performance can be significantly improved and the under-translation cases of high-entropy words can be sharply reduced.

The contributions are listed as follows:

1) We thoroughly analyze the under-translation phenomenon and find that source words with larger translation entropy are more likely to be dropped by the neural model.

2) We propose a coarse-to-fine framework to address the under-translation problem of high-entropy words. In coarse-grained phase, we construct the pseudo target sentences to reduce the entropy of these high-entropy words. In fine-grained phase, the derived pseudo target sentences are utilized to boost the original NMT model.

Neural Machine Translation

NMT contains an encoder and a decoder. The encoder transforms a source sentence $X = \{x_1, x_2, ..., x_{Tx}\}$ into a set of context vectors $C = (h_1^S, h_2^S, ..., h_{Tx}^S)$ by the LSTM (Hochreiter and Schmidhuber 1997) layers (Bahdanau, Cho, and Bengio 2015), convolutional networks (Gehring et al. 2017) or self attention mechanism (Vaswani et al. 2017).

The decoder generates one target word at a time by computing $p_T(y_i|y_{<i}, c_i)$ as follows:

$$p_T(y_i|y_{<i}, c_i) = softmax(E_{y_i}\widetilde{h_i^T} + b_s) \quad (1)$$

where E_{y_i} is the embedding of the target word y_i, and $\widetilde{h_i^T}$ is the attention output: $\widetilde{h_i^T} = tanh(W_a[h_i^T; c_i])$, where c_i is the context embedding and can be calculated as follows:

$$c_i = \sum_{j=1}^{Tx} a_{ij} h_j^S \quad (2)$$

where $a_{i,j}$ is the attention weight.

$$a_{i,j} = \frac{h_j^S h_i^T}{\sum_j h_j^S h_i^T} \quad (3)$$

where h_i^T is the hidden state in decoder. More detailed introduction can be found in (Luong, Pham, and Manning 2015) and (Vaswani et al. 2017).

Notation In this paper, we denote the whole source vocabulary by $V_S = \{s_m\}_{m=1}^{|V_S|}$ and target vocabulary by $V_T = \{t_n\}_{n=1}^{|V_T|}$, where s_m is the source word and t_n is the target word. We denote a source sentence by X and a target sentence by Y. Each source word in X is denoted by x_j. Each target word in Y is denoted by y_i. Accordingly, a target word can be denoted not only by t_n, but also by y_i. This does not contradict. t_n means this target word is the n^{th} word in vocabulary V_T, and y_i means this target word is the i^{th} word in sentence Y. Similarly, we denote a source word by s_m and x_j.

Observation and Motivation

We are curious about the following question: what kinds of source words are more likely to be ignored by the NMT model? To answer this question, we train the following three baseline NMT systems by using a Chinese-to-English dataset which contains 2.1M sentence pairs:

1) **RNMT(word)**: The baseline NMT system using two LSTM layers as encoder and decoder.

2) **RNMT(sub-word)**: Similar to **RNMT(word)**, except that we use the sub-word (Sennrich, Haddow, and Birch 2016) as the translation unit.

3) **Transformer(sub-word)**: The state-of-the-art NMT system with the self-attention mechanism (Vaswani et al. 2017).

Then we need to find out which source words are missed by the neural model. Here, we conduct the manual analysis and automatic analysis as follows:

1) **Manual Analysis:** We randomly select 800 sentences produced by the NMT model and ask five annotators to label the missed source content words.

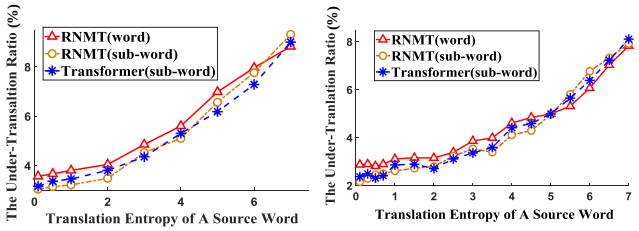
2) **Automatic Analysis:** We automatically detect the missed source content words by using the alignment tool. Specifically, given N source sentences and their corresponding translation results produced by the NMT model, we can get the alignments by using the alignment tool. If a source content word[1] has no target word to be aligned to, we treat this word being dropped by the neural model. we use 400K sentences in automatic analysis.

We analyze the relationship between various factors of source words and the under-translation ratio. Among them, we observe that **translation entropy** heavily affects the under-translation ratio. According to the manual analysis, 42.2% under-translation cases occur on the high-entropy words, which take the largest proportion in all under-translation cases.

The translation entropy can measure the translation uncertainty of a word and its formal definition is as follows:

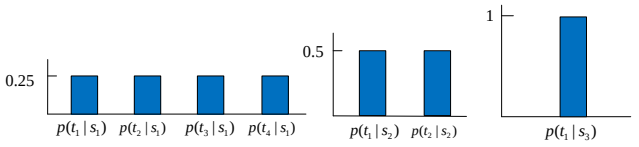
Definition (translation entropy): Assume a word s contains K candidate translations, each of which has a probability p_k, the translation entropy for this word can be calculated

[1]Words consist of content words and function words. For simplicity, we treat the most frequent 500 words as the function words.



(a) manual analysis (b) automatic analysis

Figure 2: The relationship between the ratio of under-translation (%) (y axis) and translation entropy (x axis), where (a) and (b) report the results of manual analysis and automatic analysis, respectively.



(a) word s_1 (b) word s_2 (c) word s_3

Figure 3: A toy example to illustrate the observation. The translation entropy are ranked by $E(s_1) > E(s_2) > E(s_3)$.

by

$$E(s) = -\sum_{k=1}^{K} p_k * log\ p_k \quad (4)$$

Fig. 2 reports the relationship between the ratio of under-translation (y axis) and translation entropy (x axis), where (a) and (b) show the results of manual analysis and automatic analysis, respectively. As shown in the figure, both manual analysis and automatic analysis show that in all three NMT systems (RNMT(word), RNMT(sub-word) and Transformer(sub-word)), the ratio of under-translation becomes larger with the translation entropy increases. Therefore, we can empirically obtain the following observation:

Observation: For a source word s, the larger its translation entropy is, the more likely this word is to be ignored by the neural model.

Example Fig. 3 shows a toy example to illustrate the observation. In the example there are three source words s_1, s_2 and s_3, whose translation probabilities are shown in the figure. Specifically, word s_1 has four candidate translations t_1, t_2, t_3 and t_4, and the translation probabilities are all 0.25. Word s_2 can be translated into two candidate words t_1 and t_2, and the probabilities are both 0.5. Word s_3 can only be translated to t_1. According to Eq. (4), the translation entropy can be ranked by $E(s_1) > E(s_2) > E(s_3)$. Therefore, s_1 is most likely to be missed by the NMT, s_2 is the next, and s_3 is the least. As shown in the figure, if a source word contains more candidates with a uniform distribution, its translation entropy is larger, consequently it is more likely to be ignored by the neural model.

This observation can provide a possible explanation for the phenomenon in Fig. 1. The entropy for "jia@@ (false)", "e@@ (bad)" are respectively 5.84 and 6.02, which are much higher than the entropy of "chou (ugly)" (4.26). That may lead to the phenomena that sub-words "jia@@ (false)"and "e@@ (bad)" are dropped and the "chou (ugly)" is correctly translated.

High-entropy Words Given the training bilingual dataset $D_{xy} = \{X^{(n)}, Y^{(n)}\}_{n=1}^{N}$, we can get a lexicon translation table through statistical methods (Koehn et al. 2007) and then calculate the translation entropy for each source word. If the translation entropy of a source word s exceeds the pre-defined threshold e_0, i.e., $E(s) > e_0$, we treat this word as a **high-entropy word**. According to the observation, these words have larger probabilities to be ignored than others. Therefore, our goal in this paper is to reduce the under-translation cases of these high-entropy words[2].

Method Description

To address the under-translation problem of high-entropy words, we propose a coarse-to-fine framework. In coarse-grained phase, we construct the pseudo target sentences to reduce the entropy. In fine-grained phase, the derived pseudo sentences are utilized to improve the neural model.

Coarse-grained Phase

In coarse-grained phase, we construct the pseudo target sentences to reduce the entropy of these high-entropy words. Here, we utilize an example in Fig. 4 to introduce our construction method. In the example, word s_1 is a high-entropy word, and contains four candidates t_1, t_2, t_3 and t_4 with a uniform probability distribution. Assume the training bilingual dataset D_{xy} contains the following three pairs: In m^{th} pair (X^m, Y^m), s_1 should to be translated into t_1. In n^{th} pair (X^n, Y^n), t_2 is the correct translation for s_1. In p^{th} pair (X^p, Y^p), s_1 needs to be translated into t_3 and t_4 simultaneously. Fig. 4(a) shows our construction method. In pseudo target sentences, we replace these candidate words t_1, t_2, t_3 and t_4 by a special pseudo token $stoken4s_1$ and keep the other target words unchanged. By doing so, the probability distribution will change as illustrated in Fig 4(b). Meanwhile, the entropy of s_1 can be reduced sharply.

Assuming there are M high-entropy words[3] $\{s_m\}_{m=1}^{M}$, we first generate M special pseudo tokens $\{stoken4s_m\}_{m=1}^{M}$. Then we construct the pseudo target sentences by replacing all candidate translations of s_m with the corresponding special token $stoken4s_m$.

[2]We think the reason why these high-entropy words are more likely to be dropped may due to their high translation uncertainty. On the one hand, as high-entropy words contain various candidate translations, it is more difficult for the neural model to learn their alignments, which may lead to more under-translation cases of these words. On the other hands, the task of the beam search decoding is to find the full sentence translation with the highest probability. Generally, the candidate translations of a high-entropy word always have lower probabilities as shown in Fig. (3), making these candidate translations more likely be dropped during the beam search decoding.

[3]The number M is determined by the pre-defined entropy threshold e_0.

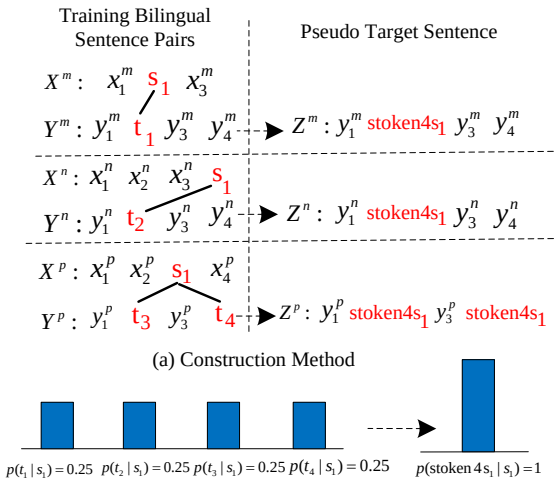


Figure 4: Illustration of the construction method. (a) We construct the pseudo target sentences by replacing t_1, t_2, t_3 and t_4 by $stoken4s_1$. (b) shows the probability distribution changing.

Note that the other target words remain unchanged. After this step, compared to the original bilingual dataset $D_{xy} = \{X^{(n)}, Y^{(n)}\}_{n=1}^{N}$, now we can get a tri-lingual dataset $D_{xyz} = \{X^{(n)}, Y^{(n)}, Z^{(n)}\}_{n=1}^{N}$, where X and Y are the original source and target sentences. Z is the derived pseudo target sentence.

Fine-grained Phase

Compared to the original target sentence Y, the derived pseudo target sentence Z is coarse-grained, which decreases the entropy of high-entropy words and also reduces the translation difficulty. Now our task is to improve the fine-grained neural model with pseudo sentences Z. Here we propose three methods as follows:

Pre-training Method Fig. 5(a) shows our first method, in which we utilize the derived pseudo sentences to provide a better parameter initiation. To achieve this, we first pre-train the model θ by using the data set $D_{xz} = \{X^{(n)}, Z^{(n)}\}_{n=1}^{N}$, then fine-tune it by the original data D_{xy}. Compared to the original method which initializes θ randomly, we hope the pre-trained model could provide the better initiation.

Multitask Method Fig. 5(b) illustrates the framework of the multitask method, in which we simultaneously train the neural model θ through two translation tasks: 1) the task from X to Y, and 2) the task from X to Z. To achieve this, the object function is redesigned by

$$L(\theta, D_{xyz}) = \sum_{n=1}^{N}\sum_{i}^{T_y}(log\ p_T(y_i^{(n)}|X^{(n)}, \theta) + \lambda * log\ p_T(z_i^{(n)}|X^{(n)}, \theta)) \quad (5)$$

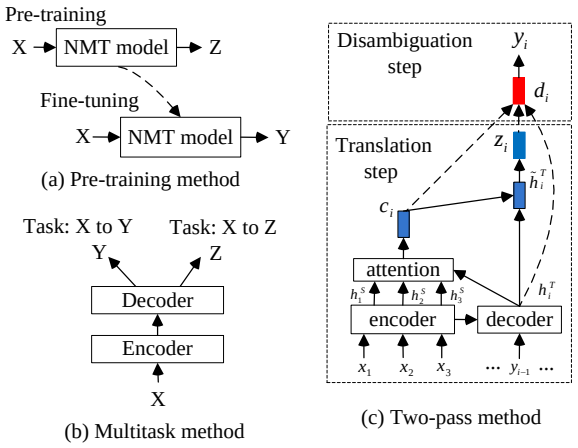


Figure 5: The proposed three methods in fine-grained phase: (a) pre-training method, (b) multitask method and (c) two-pass method.

where λ is pre-defined weight to balance the two tasks. As shown in Eq. (5), we utilize the pseudo translation task as an auxiliary task to encourage the neural model focus more on the high-entropy words.

Two-pass Method In the two-pass method, we divide the whole translation process into two steps: **translation step** and **disambiguation step** as follows:

In **translation step**, we translate source sentence X to pseudo target sentence Z through the standard NMT framework. As the pseudo sentences could reduce the entropy of these high-entropy words, in this step, we hope these words could be translated to the special pseudo tokens.

After this step, we can get a predicted translation distribution $p_T(z_i|X)$, which can be calculated by the standard NMT framework in Eq. (1). The parameters in this step are denoted by θ_T.

In **disambiguation step**, we need to transform the special token in Z to real target word in Y. To achieve this, two cases must be considered:

Case when z_i is a normal token. In this case, as z_i is a normal token and same as y_i, thus $p_D(y_i|X)$ is directly same as $p_T(z_i|X)$, i.e., $p_D(y_i|X) = p_T(z_i|X)$, where $p_D(.)$ denotes the disambiguation probability distribution.

Case when z_i is a special token. In this case, we design a disambiguation network to transform $p_T(z_i|X)$ into $p_D(y_i|X)$.

As shown in Fig 5(c), the core of the disambiguation network is the disambiguation vector d_i, which can be calculated by

$$\begin{aligned} d_i &= f(E_{z_i}, c_i, h_i^T) \\ &= tanh(W_z * E_{z_i} + W_c * c_i + W_h * h_i^T + b_d) \end{aligned} \quad (6)$$

where W_z, W_c, W_h and b_d are parameters, which are denoted by θ_D. From Eq. (6), the disambiguation vector d_i is determined by three factors: 1) The embedding of special token E_{z_i}. 2) The context vector c_i in Eq. (2). It is obviously to

consider the context information to decide the correct translation. 3) The hidden states of decoder h_i^T.

The next task is to calculate the assignment score $s_{(z_i,y_i)}$. Its function is transforming the translation probability of special token $p_T(z_i|X)$ into the disambiguation probability of real gold word $p_D(y_i|X)$. Here, $s_{(z_i,y_i)}$ can be calculated by

$$s_{(z_i,y_i)} = softmax(E_{y_i} * d_i) \quad (7)$$

where E_{y_i} is the embedding of gold target word y_i. d_i is the disambiguation vector in Eq. (6).

Finally, we can get the final disambiguation probability distribution by

$$p_D(y_i|X) = p_T(z_i|X) * s_{(z_i,y_i)} \quad (8)$$

The parameters in these two steps (θ_T and θ_D) can be optimized through an end-to-end manner with the following object function:

$$L(\theta_D, \theta_T, D_{xyz}) = \sum_{n=1}^{N} \sum_{i}^{T_y} (log\ p_D(y_i^{(n)}|X^{(n)}, \theta_D) + \lambda * log\ p_T(z_i^{(n)}|X^{(n)}, \theta_T)) \quad (9)$$

where λ is the pre-defined weight to balance the two steps.

Experimental Settings

We test the proposed methods on Chinese-to-English (CH-EN), English-to-Japanese (EN-JA) and English-to-German (EN-DE) translation. In CH-EN translation, we use LDC corpus which includes 2.1M sentence pairs for training. NIST 2003 dataset is used for validation. NIST04-06 and 08 datasets are used for testing. In EN-JA translation, we use KFTT dataset[4], which includes 0.44M sentence pairs for training, 1166 sentence pairs for validation and 1160 sentence pairs for testing. In EN-DE translation, we use WMT 2014 EN-DE dataset, which includes 4.5M sentence pairs for training. 2012-2013 datasets are used for validation and 2014 dataset is used for testing. We test all methods based on two granularities: words and sub-words. For word granularity, we limit the vocabulary to 30K (CH-EN), 30K (EN-JA) and 50K (EN-DE) for both the source and target languages. For sub-word granularity, we use the BPE method (Sennrich, Haddow, and Birch 2016) to merge 30K (CH-EN), 30K (EN-JA) and 37K (EN-DE) steps. In CH-EN and EN-JA translation, we use case-insensitive 4-gram BLEU (Papineni et al. 2002) for translation quality evaluation. In EN-DE translation, we use case-sensitive 4-gram BLEU for evaluation.

We use the fast-align tool (Dyer, Chahuneau, and Smith 2013) with both source-to-target and target-to-source directions to extract the word alignments.

We compare our method with other relevant methods as follows:

1) **RNMT**: The baseline NMT system using two LSTM layers as encoder and decoder. The word embedding dimension and the size of hidden layers are both set to 1,000. The minibatch size is set to 128 (Zoph and Knight 2016)[5].

[4]http://www.phontron.com/kftt/.

[5]https://github.com/isi-nlp/Zoph_RNN.

source (word): 最终 真善美 彻底 打败 了 假恶丑
source (sub-word): 最终 真@@ 善@@ 美 彻底 打败 了 **假@@ 恶@@ 丑**
pinyin: (sub-word): zuizhong zhen@@ shan@@ mei chedi dabai le **jia@@ e@@ chou**
reference: finally , the true , the good and the beautiful completely defeated **the false**, **the bad** and **the ugly**
RNMT: eventually , true , good , and beauty thoroughly defeated the ugly
RNMT+two_pass: finally, true, good and beauty defeated thoroughly **the false**, **the evil** and **the ugly**

Figure 6: The two sub-words "jia@@" and "e@@" are correctly translated by our method. While they are missed by the NMT model.

2) **RNMT+coverage**: The coverage model proposed in (Tu et al. 2016), we implement this model on the basis of RNMT, in which we select linguistic coverage model to construct the coverage vector.

3) **Transformer**: The state-of-the-art NMT system with self-attention mechanism. The hyper-parameters are the same as (Vaswani et al. 2017)[6].

4) **X+pre_train**, **X+multitask**, and **X+two_pass**: These are our proposed methods. In all methods, the entropy threshold $e_0 = 4$. In the pre-training method, we first pre-train the model 10 epochs with the pseudo sentences. In the multitask method, the balance weight λ in Eq. (5) is set to 0.35. In the two-pass method, the balance weight λ in Eq. (9) is set to 0.3. All these hyper-parameters are fine-tuned on the validation set.

Results on CH-EN Translation

Translation Quality Analysis

Results on RNMT Model Table 1 reports the main translation results of CH-EN translation. We first compare our method with RNMT. As shown in row 1 and row 2-4 in Table 1, our methods can improve over RNMT on all test datasets, where the RNMT+two_pass achieves the best performance and the improvement reaches to 1.21 BLEU (40.37 vs. 39.16) points.

We also test the proposed methods when the translation unit is sub-word. The results are shown in row 5-8. As shown in the table, the improvement of RNMT+two_pass is also the largest and the gains can reach to 0.95 BLEU points (42.29 vs. 41.34). RNMT+multitask is the second and can improve the RNMT by 0.64 BLEU (41.98 vs. 41.34).

Fig. 6 shows the mentioned example, in which RNMT missed two source sub-words "jia@@" and "e@@". While in the proposed two-pass method, these two sub-words can be correctly translated.

Effect of the Hyper-parameters Generally, the two-pass method achieves the best result. As we discussed before, there are two hyper-parameters in our two-pass method, i.e., 1) the entropy threshold e_0 and 2) the balance weight λ in Eq. (9). Table 2 reports the BLEU scores under different

[6]https://github.com/tensorflow/tensor2tensor.

#	Model	Units	03	04	05	06	08	Avg.
1	RNMT	word	41.01	42.94	40.31	40.57	30.96	39.16
2	RNMT+pre_train	word	41.53†	43.46*	40.41*	41.35†	31.17*	39.58
3	RNMT+multitask	word	41.99†	43.95†	40.84†	41.57†	31.42*	39.95
4	RNMT+two_pass	word	**42.37**†	**44.27**†	**41.58**†	**41.72**†	**31.91**†	**40.37**
5	RNMT	sub-word	43.96	44.74	42.46	43.01	32.53	41.34
6	RNMT+pre_train	sub-word	44.13	44.96*	42.61*	43.31*	32.77*	41.56
7	RNMT+multitask	sub-word	44.53†	45.17*	**43.39**†	43.85†	32.97*	41.98
8	RNMT+two_pass	sub-word	**44.86**†	**45.64**†	43.36†	**44.17**†	**33.44**†	**42.29**
9	RNMT+coverage	word	41.75	43.79	41.44	41.24	31.46	39.94
10	RNMT+coverage+multitask	word	42.66†	**44.54**†	42.07†	41.77†	32.03†	40.61
11	RNMT+coverage+two_pass	word	**42.94**†	44.52†	**42.53**†	**42.12**†	**32.23**†	**40.87**
12	Transformer	sub-word	45.80	47.77	46.90	46.90	34.61	44.40
13	Transformer+multitask	sub-word	**46.71**†	48.13*	47.41†	47.44†	34.98*	44.93
14	Transformer+two_pass	sub-word	46.64†	**48.29**†	**47.63**†	**47.51**†	**35.13**†	**45.04**

Table 1: The main results of CH-EN translation. "*" indicates that the proposed system is statistically significant better ($p < 0.05$) than the baseline system and "†" indicates $p < 0.01$.

λ \ e_0	2	4	6	8
0.1	40.24	40.24	39.63	38.99
0.2	40.27	40.31	39.77	39.76
0.3	40.30	**40.37**	39.92	39.72
0.4	40.11	40.25	39.81	39.51

Table 2: The BLEU points of two-pass methods under different hyper-parameters, where λ is the balance weight and e_0 is the entropy threshold.

hyper-parameters. As shown in the table, when $\lambda = 0.3$ and $e_0 = 4$, our two-pass method achieves the best results. An interesting thing should be noticed is that when e_0 reduces from 8 to 4, more words are replaced with the special tokens, and the performance increases. While when e_0 reduces to 2, the performance no longer increases. We think the reason is that the neural model can correctly translate these low-entropy words.

Results on Coverage Model We also compare the proposed method with the coverage model. The results are shown in row 9-11. Compared to the coverage model, the two-pass method can outperform the coverage model by 0.43 BLEU points (40.37 vs. 39.94). More importantly, on the basis of the coverage model, our method can further improve the model by 0.93 BLEU points (40.87 vs. 39.94), indicating that coverage model still faces the under-translation problem of high-entropy words, and our method alleviates this problem.

Results on Self-attention Model We conduct experiments to evaluate the performance of proposed method on the basis of self-attention model (Transformer). As shown in row 12-14 of Table 1, our method can also improve the translation quality on Transformer, where the two-pass method can boost the Transformer by 0.64 BLEU points and the improvement for the multitask method is 0.53 BLEU points.

Model	All	High-Entropy	Other
RNMT	5.02%(207)	8.05%(91)	3.88%(116)
+two_pass	4.10%(169)	4.86%(55)	3.81%(114)
Coverage	4.32%(178)	7.07%(80)	3.28%(98)
+two_pass	3.59%(148)	4.77%(54)	3.14%(94)
Transformer	4.63%(191)	7.60%(86)	3.51%(105)
+two_pass	3.78%(156)	4.69%(53)	3.44%(103)

Table 3: The under-translation ratio (number) of different methods. Column **All**, **High-Entropy** and **Other** list the total missed ratio (number) of all words, high-entropy words, and the other words, respectively.

Under-translation Analysis

Statistics of Under-translation As our method tends to reduce the under-translation cases of high-entropy words. Therefore, we also conduct a manual analysis to figure out how many dropped high-entropy words could be rectified by our methods. We randomly select 200 testing sentences, and count the following three numbers: 1) the ratio (number) of all dropped words (**All**), 2) the ratio (number) of dropped high-entropy words (**High-Entropy**), and 3) the ratio (number) of dropped other words (**Other**). The statistics are reported in Table 3. From this table, we can reach the following three conclusions:

1) On the basis of RNMT model, our method (+two_pass) can reduce the under-translation ratio (number) of high-entropy words from 8.05% (91 times) to 4.86% (55 times). As a comparison, our method has less effect on the under-translation of other words (3.88% (116 times)) vs. 3.81% (114 times)), showing that our method is the most effective to reduce the under-translation cases of high-entropy words.

2) Coverage model can both reduce the under-translation cases of high-entropy words (8.05% vs. 7.07%) and other words (3.88% vs. 3.28%). On the basis of Coverage model, our method could further reduce the under-translation ratio of high-entropy words to 4.77%, which shows that our methods and coverage model are complementary.

Model	All	High-Entropy
RNMT	45.54	51.34
RNMT+ multitask	43.39	47.18
RNMT+ two_pass	42.21	46.37
Transformer	44.31	49.47
Transformer +multitask	41.41	45.11
Transformer +two_pass	40.22	44.02

Table 4: The alignment error rate (AER) of different methods. The lower the score is, the better the alignment quality is. Column **All** and **High-Entropy** list the AER of all words and high-entropy words, respectively.

Model	EN-JA	EN-DE
RNMT	27.43	23.85
RNMT+multitask	28.21†	24.66†
RNMT+two_pass	28.56†	24.89†
Transformer	29.50	27.20
Transformer+multitask	29.93*	27.59*
Transformer+two_pass	30.28†	27.73†

Table 5: The results on EN-JA and EN-DE translation. "*" indicates that it is statistically significantly better ($p < 0.05$) than baseline system and "†" indicates $p < 0.01$.

3) On Transformer model, we can get similar conclusions that our method can sharply reduce the under-translation ratio (times) of high-entropy words from 7.60% (86 times) to 4.69% (53 times).

Alignment Quality We also carry out another experiment to analyze the alignment quality of the high-entropy words. The evaluation dataset is from (Liu and Sun 2015), which contains 900 manually aligned Chinese-English sentence pairs. We utilize the alignment error rate (AER) (Och and Ney 2003) to measure the performance. The lower the score is, the better the alignment quality is. Table 4 lists the alignment performance. As shown in this table, on both the RNMT and Transformer models, our method could improve the alignment quality, especially for the high-entropy words.

Results on EN-JA and EN-DE Translation

We also test our method on EN-JA translation and EN-DE translation. The results are reported in Table 5, our method improves the translation quality on both EN-JA and EN-DE translation. Specifically, on EN-JA translation, the proposed two-pass method outperforms the baseline model by 1.13 BLEU (RNMT) and 0.78 BLEU points (Transformer), respectively. On EN-DE translation, the two-pass method reaches 1.04 BLEU points (RNMT) and 0.53 BLEU points improvement (Transformer).

Related Work

The related work can be divided into three categories and we describe each of them as follows:

Addressing the Under-translation Several studies address the under-translation problem from the model perspective. For example, Tu et al. (2016) and Mi et al. (2016) borrow the coverage mechanism to improve the attention mechanism. In their models, they maintain a coverage vector to collect the attention record, then use this coverage vector to adjust the attention in next time step. Some studies tend to improve the hidden states of encoder and decoder, where Tu et al. (2017) propose a reconstructor to ensure the information in encoder can being adequately transformed to decoder. Zheng et al. (2018) extend the NMT model to optimize the hidden states of encoder and decoder by modeling past and future vectors. Different from above methods, we tackle this problem from the entropy perspective. We observe that a source word with a larger translation entropy is more likely to be ignored and propose a coarse-to-fine framework to address this problem. More importantly, the experiments show that our methods could further reduce the under-translation cases on the basis of previous methods.

Addressing the Ambiguous Words As the high-entropy words are always the ambiguous words. Our work is also related to the studies dealing with the ambiguous words, e.g., (Vickrey et al. 2005; Carpuat and Wu 2007; Rios, Mascarell, and Sennrich 2017; Sennrich 2017). The goal of these methods is to improve the translation accuracy of ambiguous words. In contrast, we try to address the under-translation problem of these words.

Combining SMT and NMT Our method utilizes the lexicon translation table and is also related to the work which utilizes lexicon and phrase translation table into NMT, e.g., (Arthur, Neubig, and Nakamura 2016; Tang et al. 2016; Feng et al. 2017; Dahlmann et al. 2017; Wang et al. 2017a; Zhang and Zong 2016; Zhou et al. 2017; Wang et al. 2017b; Zhao et al. 2018a; Huang et al. 2018; Zhao et al. 2018b). These methods tend to improve the translation quality for the rare words, while our methods focus on the under-translation problem of high-entropy words.

Conclusions

Through error analysis, we find that source words with larger translation entropy are more likely to be dropped. To address this problem, we propose a coarse-to-fine framework. In coarse-grained phase, we construct the pseudo target sentences to reduce the entropy of the high-entropy words. In fine-grained phase, we propose three methods to reduce the under-translation cases with derived pseudo sentences. The extensive experiments demonstrate that our method significantly outperforms the strong baseline models in translation quality and sharply reduces the under-translation cases of these high-entropy words.

Acknowledgments

The research work described in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2017YFC0822505 and the Natural Science Foundation of China under Grant No. 61333018. The research work in this paper also has been supported by Beijing Advanced Innovation for Language Resources of Beijing Language and Culture University.

References

Arthur, P.; Neubig, G.; and Nakamura, S. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of EMNLP 2016*, 1557–1567.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Carpuat, M., and Wu, D. 2007. Improving statistical machine translation using word sense disambiguation. In *proceedings of EMNLP 2007*, 61–72.

Cho, K.; Gulcehre, B. v. M. C.; Bahdanau, D.; Schwenk, F. B. H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, 1724–1734.

Dahlmann, L.; Matusov, E.; Petrushkov, P.; and Khadivi, S. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. In *Proceedings of EMNLP 2017*, 1422–1431.

Dyer, C.; Chahuneau, V.; and Smith, N. A. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL 2013*.

Feng, Y.; Zhang, S.; Zhang, A.; Wang, D.; and Abel, A. 2017. Memory-augmented neural machine translation. In *Proceedings of EMNLP 2017*, 1401–1410.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1601.03317*.

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

Huang, P. S.; Wang, C.; Huang, S.; Zhou, D.; and Deng, L. 2018. Towards neural phrase-based machine translation. In *Proceedings of ICLR 2018*.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP 2013*, 1700–1709.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007*, 177–180.

Liu, Y., and Sun, M. 2015. Contrastive unsupervised word alignment with non-local features. In *proceedings of AAAI 2015*, 2295–2301.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*, 1412–1421.

Mi, H.; Sankaran, B.; Wang, Z.; and Ittycheriah, A. 2016. Coverage embedding model for neural machine translation. In *Proceedings of EMNLP 2016*, 955–960.

Och, F. J., and Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 19–51.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, 311–318.

Rios, A.; Mascarell, L.; and Sennrich, R. 2017. Improving word sense disambiguation in neural machine translation with sense embeddings. In *proceedings of WMT 2017*.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*, 1715–1725.

Sennrich, R. 2017. How grammatical is character-level neural machine translation? assessing mt quality with contrastive translation pairs. In *Proceedings of EACL 2017*, 376–382.

Tang, Y.; Meng, F.; Lu, Z.; Li, H.; and Yu, P. L. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Tu, Z.; Lu, Z.; Liu, Y.; Liu, X.; and Li, H. 2016. Coverage-based neural machine translation. In *Proceedings of ACL 2016*, 76–85.

Tu, Z.; Liu, Y.; Shang, L.; Liu, X.; and Li, H. 2017. Neural machine translation with reconstruction. In *Proceedings of AAAI 2017*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; and Kaiser, u. 2017. Attention is all you need. *arXiv preprint arXiv:1601.03317*.

Vickrey, D.; Biewald, L.; Teyssier, M.; and Koller, D. 2005. Word-sense disambiguation for machine translation. In *proceedings of ACL 2005*, 771–778.

Wang, X.; Lu, Z.; Tu, Z.; Li, H.; Xiong, D.; and Zhang, M. 2017a. Neural machine translation advised by statistical machine translation. In *Proceedings of AAAI 2017*.

Wang, Y.; Zhao, Y.; Zhang, J.; Zong, C.; and Xue, Z. 2017b. Towards neural machine translation with partially aligned corpora. In *Proceedings of IJCNLP 2017*.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhang, J., and Zong, C. 2016. Bridging neural machine translation and bilingual dictionaries. *arXiv preprint arXiv:1610.07272*.

Zhao, Y.; Wang, Y.; Zhang, J.; and Zong, C. 2018a. Phrase table as recommendation memory for neural machine translation. In *Proceedings of IJCAI 2018*, 4609–4615.

Zhao, Y.; Zhang, J.; He, Z.; Zong, C.; and Wu, H. 2018b. Addressing troublesome words in neural machine translation. In *Proceedings of EMNLP 2018*, 391–400.

Zheng, Z.; Zhou, H.; Huang, S.; Mou, L.; Dai, X.; Chen, J.; and Tu, Z. 2018. Modeling past and future for neural machine translation. *TACL* 145–157.

Zhou, L.; Hu, W.; Zhang, J.; and Zong, C. 2017. Neural system combination for machine translation. In *Proceedings of ACL 2017*, 378–384.

Zoph, B., and Knight, K. 2016. Multi-source neural translation. In *Proceedings of NAACL 2016*, 30–34.