Adaptive Weighting for Neural Machine Translation

Yachao Li[1,2], **Junhui Li**[1], **Min Zhang**[1,*]
[1]Institute of Artificial Intelligence,
School of Computer Science and Technology, Soochow University, China
[2]Key Laboratory of China's Ethnic Languages and Information Technology
of Ministry of Education, Northwest Minzu University, China
liyc7711@gmail.com, {lijunhui, minzhang}@suda.edu.cn

Abstract

In the popular sequence to sequence (seq2seq) neural machine translation (NMT), there exist many weighted sum models (WSMs), each of which takes a set of input and generates one output. However, the weights in a WSM are independent of each other and fixed for all inputs, suggesting that by ignoring different needs of inputs, the WSM lacks effective control on the influence of each input. In this paper, we propose adaptive weighting for WSMs to control the contribution of each input. Specifically, we apply adaptive weighting for both GRU and the output state in NMT. Experimentation on Chinese-to-English translation and English-to-German translation demonstrates that the proposed adaptive weighting is able to much improve translation accuracy by achieving significant improvement of 1.49 and 0.92 BLEU points for the two translation tasks. Moreover, we discuss in-depth on what type of information is encoded in the encoder and how information influences the generation of target words in the decoder.

1 Introduction

Recent advances in neural machine translation (NMT) have achieved remarkable success over the state-of-the-art of statistical machine translation (SMT) on various language pairs (Bahdanau et al., 2015; Jean et al., 2015; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017). In the neural networks of seq2seq models, either RNN-based (Bahdanau et al., 2015), CNN-based (Gehring et al., 2017), or full attention-based (Vaswani et al., 2017), there exist many scenarios in both encoder and decoder where a weighted sum model (WSM) takes a set of inputs and generate one output. As shown in Eq. 1, the WSM first combines k inputs $(x_1, \cdots, x_k)$ with k respective weights $(w_1, \cdots, w_k)$ and then non-linearizes it through an activation function f, such as $\tanh$, *sigmoid function*, $ReLU$, and so on. In this paper we omit bias terms to make the equations less cluttered.

$$o = f\left(\sum_{i=1}^{k} w_i x_i\right) \tag{1}$$

Note that the above weights $(w_1, \cdots, w_k)$ are independent of each other and once the model is tuned, the weights are fixed for all inputs, suggesting that by ignoring different needs of the inputs, the WSM lacks effective control on the influence of each input.

Let us take a concrete scenario in seq2seq model as an example. Figure 1(a) shows a typical illustration of generation of t-th target word where the decoder takes three inputs, i.e., source context c_t, previous target word y_{t-1} and current target state s_t while generating one output y_t via output state o_t. However, the study in Tu et al. (2017) suggests that different target words require inconsistent contributions from source context (c_t) and target context (i.e., y_{t-1} and s_t). For example, to generate translation *Xinhua News Agency , Hong Kong*, the first word y_1 *xinhua* is highly related to its source context c_1 while the

*Min Zhang is Corresponding Author.

Proceedings of the 27th International Conference on Computational Linguistics, pages 3038–3048
Santa Fe, New Mexico, USA, August 20-26, 2018.

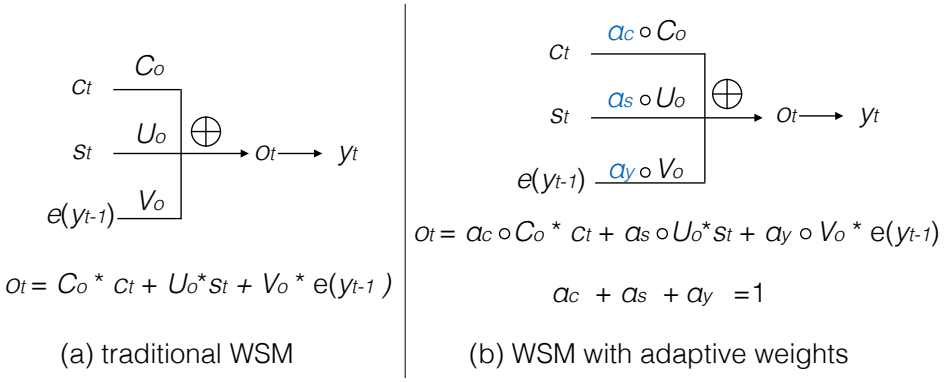

Figure 1: Illustration of generation of t-th translation word.

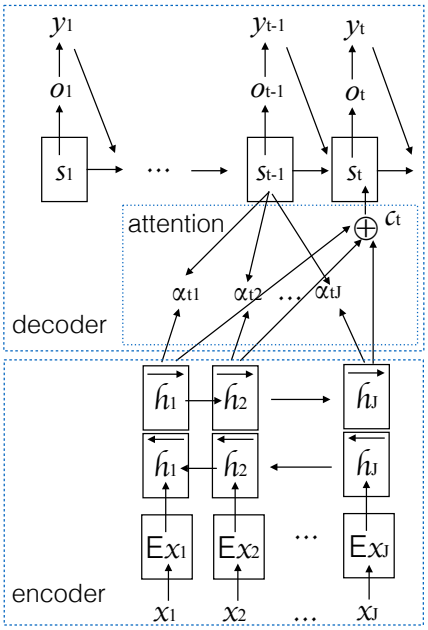

Figure 2: NMT model with attention mechanism.

second and third words y_2 *news* and y_3 *agency* are mainly influenced by target context due to the well-formed saying *Xinhua News Agency*. Similarly, y_5 *Hong* and y_6 *Kong* are mainly influenced by source context and target context, respectively.

In this paper, we propose adaptive weighting to dynamically control the contribution of each input in a WSM that has a set of inputs. Unlike the conventional weights that are independent of each other, adaptive weights are learned to be dependent on each other and more importantly be able to dynamically select the amount of input information. Specifically, we use gate mechanism to incorporate adaptive weights in GRU and in computing the output states. Experimentation on both Chinese-to-English and English-to-German translation tasks demonstrates that NMT systems with adaptive weighting are able to much improve the translation accuracy. Moreover, through adaptive weights we discuss in-depth on what type of information encoded in the encoder and how information influences the generation of a target word.

2 NMT with Attention Mechanism

In this section, we review NMT model with attention mechanism. Encoder-decoder model with attention mechanism is one of the most popular frameworks for NMT (Bahdanau et al., 2015), which consists of

an encoder and a decoder, as shown in Figure 2. In the following, m refers to the embedding size of both source and target sides, and n the hidden state size of the two sides. E_{x_j} returns the word embedding for source word x_j while $e(y_t)$ returns the word embedding for target word y_t.

Encoder: Given a source sentence $\mathbf{x} = (x_1, \cdots, x_J)$, the encoder first converts each word x_j into a real-valued m-dimensional vector E_{x_j} via the embedding matrix $E \in \mathbb{R}^{m \times |V_s|}$, where V_s is the source-side vocabulary. Then the encoder uses bidirectional RNN: the forward RNN reads the input sequence from left to right and outputs a forward sequence of hidden states $(\overrightarrow{h_1}, \cdots, \overrightarrow{h_J})$ by $\overrightarrow{h_j} = RNN(\overrightarrow{h_{j-1}}, E_{x_j})$; likewise the backward RNN operates from right to left and outputs a backward sequence $(\overleftarrow{h_1}, \cdots, \overleftarrow{h_J})$. Each source word x_j is represented as h_j (also referred to as word annotation vector): the concatenation of hidden states $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$. Such bidirectional RNN encodes not only the word itself but also its left and right context, which can provide important evidence for its translation.

Decoder: The decoder is also an RNN that predicts a target sequence $\mathbf{y} = (y_1, \cdots, y_T)$. It defines a probability over the translation $\mathbf{y}$ by decomposing the joint probability into the ordered conditionals:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | y_1, \cdots, y_{t-1}, \mathbf{x}) \tag{2}$$

$$p(y_t | y_1, \cdots, y_{t-1}, \mathbf{x}) = softmax\left(\mathbf{W}_o f(o_t)\right) \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{|V_t| \times m}$ is a weight matrix, and V_t is the target-side vocabulary. o_t is the **output state**, defined as:

$$o_t = \mathbf{U}_o s_t + \mathbf{V}_o e(y_{t-1}) + \mathbf{C}_o c_t \tag{4}$$

$$s_t = RNN(s_{t-1}, e(y_{t-1}), c_t) \tag{5}$$

where $\mathbf{U}_o \in \mathbb{R}^{n \times n}$, $\mathbf{V}_o \in \mathbb{R}^{n \times m}$, $\mathbf{C}_o \in \mathbb{R}^{n \times 2n}$ are weight matrices. s_t is the hidden state of the decoder. c_t is the context vector, which is calculated as the summation vector weighted by a_{tj}:

$$c_t = \sum_{j=1}^{J} a_{tj} h_j \tag{6}$$

where a_{tj} is the weight of h_j, defined as

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{J} exp(e_{tk})} \tag{7}$$

where $e_{tj} = a(s_{t-1}, h_j)$ is an alignment model, measuring the degree of matching between the *j*-th source hidden unit h_j and the *t*-th target hidden unit s_t.

Training: The whole model is jointly trained to maximize the conditional log-likelihood of the training data containing I sentence pairs.

$$L(\Theta) = \max_{\Theta} \frac{1}{I} \sum_{i=1}^{I} \log p_\Theta(\mathbf{y}_i | \mathbf{x}_i) \tag{8}$$

where Θ is the set of all parameters, and $(\mathbf{x}_i, \mathbf{y}_i)$ is the i-th sentence pair in the training set.

3 Adaptive Weighting for NMT

In this section, we propose adaptive weighting for NMT. Our goal is to enable a WSM dynamically selects the amount of input information when there exist two or more inputs. Specifically, we apply the above idea into the attention-based seq2seq model from two perspectives: (1) adaptive weighting for GRU; and (2) adaptive weighting for the output state o_t as in Figure 1.

3.1 Gated Recurrent Unit

Gated recurrent unit (GRU) (Cho et al., 2014a; Cho et al., 2014b) is a type of hidden unit of RNN which makes each recurrent unit to adaptively capture dependencies of different time scales. A GRU has two gates called reset gate r_t and update gate z_t, which control the amount of information will "flow through" the network. r_t and z_t are computed as follows:

$$r_t = \sigma(\mathbf{W}_r Ex_t + \mathbf{U}_r h_{t-1}) \tag{9}$$

$$z_t = \sigma(\mathbf{W}_z Ex_t + \mathbf{U}_z h_{t-1}) \tag{10}$$

where $\sigma(\cdot)$ is a logistic sigmoid function. Ex_t and h_{t-1} are the input and the previous hidden state. $\mathbf{W}_r, \mathbf{W}_z \in \mathbb{R}^{n\times m}$, $\mathbf{U}_r, \mathbf{U}_z \in \mathbb{R}^{n\times n}$ are model parameters which are learned. The new hidden unit h_t is computed by

$$\tilde{h_t} = tanh(\mathbf{W}Ex_t + \mathbf{U}(r_t \circ h_{t-1})) \tag{11}$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h_t} \tag{12}$$

where $tanh(\cdot)$ is a hyperbolic tangent activation function, and $\circ$ is an element-wise multiplication. $\mathbf{W} \in \mathbb{R}^{n\times m}$, $\mathbf{U} \in \mathbb{R}^{n\times n}$ are model parameters.

3.2 Adaptive Weighting for GRU

As shown in Section 3.1, recurrent unit is critical in GRU. Both the reset gate r_t and the update gate z_t control the amount of previous state h_{t-1} and current input Ex_t being carried over to the current hidden state h_t. However, the gate itself either r_t or z_t, is uniformly set via fixed weights $\mathbf{W}_r$ (or $\mathbf{W}_z$) and $\mathbf{U}_r$ (or $\mathbf{U}_z$) for all inputs of Ex_t and all previous states of h_{t-1}. To dynamically select the amount of Ex_t and h_{t-1}, we define a hyper-gate g_t to control the flow of information between Ex_t and h_{t-1} explicitly, as shown in Eq. 13.

$$g_t = \sigma(\mathbf{W}_g Ex_t + \mathbf{U}_g h_{t-1}) \tag{13}$$

where $\mathbf{W}_g \in \mathbb{R}^{n\times m}$ and $\mathbf{U}_g \in \mathbb{R}^{n\times n}$ are parameters to be learned. Then we add g_t to the standard GRU to control how much information from previous hidden state is preserved at current time step explicitly. Accordingly, we update Eq. 9 $\sim$ 12 with the following equations.

$$r_t = \sigma((1 - g_t) \circ \mathbf{W}_r Ex_t + g_t \circ \mathbf{U}_r h_{t-1}) \tag{14}$$

$$z_t = \sigma((1 - g_t) \circ \mathbf{W}_z Ex_t + g_t \circ \mathbf{U}_z h_{t-1}) \tag{15}$$

$$\tilde{h_t} = tanh((1 - g_t) \circ \mathbf{W}Ex_t + g_t \circ \mathbf{U}(r_t \circ h_{t-1})) \tag{16}$$

$$h_t = g_t \circ z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h_t} \tag{17}$$

As shown in Eq.14, for example, the input Ex_t and the previous state h_{t-1} are now controlled by weights $(1-g_t)\circ\mathbf{W}_r$ and $g_t\circ\mathbf{U}_r$, respectively, which are dependent on each other and further controlled by the hyper-gate g_t.

Note that the NMT model adopts GRU in both encoder and decoder to generate the source side hidden states (i.e., h_t) and the target side hidden states (i.e., s_t), respectively. Therefore, we incorporate adaptive weights for GRUs in both encoder and decoder.

3.3 Adaptive Weighting for Output State

As shown in Figure 1 (b), the decoder iteratively takes three inputs, i.e., source context c_t, previous target word y_{t-1} and current target state s_t and generates one output y_t via intermediate state o_t. In order to dynamically select the amount of c_t, y_{t-1}, and s_t, we update Eq. 4 with the following:

$$o_t = \alpha_s \circ \mathbf{U}_o s_t + \alpha_y \circ \mathbf{V}_o e(y_{t-1}) + \alpha_c \circ \mathbf{C}_o c_t \tag{18}$$

where α_s, α_y, α_c can be either scalars or vectors. Together with $\mathbf{U}_o$, $\mathbf{V}_o$ and $\mathbf{C}_o$, they control the amount of s_t, y_{t-1} and c_t being carried forward, respectively. In this work, we define them as vectors, and α_s, for example, is computed as:

$$\alpha_s = \frac{exp(e_s)}{exp(e_s) + exp(e_y) + exp(e_c)} \tag{19}$$

$$e_s = f(\tilde{o}_t, s_t) \tag{20}$$

$$\tilde{o}_t = \mathbf{U}_c s_t + \mathbf{V}_c e(y_{t-1}) + \mathbf{C}_c c_t \tag{21}$$

where f is a feed-forward neural network. $\mathbf{U}_c \in \mathbb{R}^{n \times n}$, $\mathbf{V}_c \in \mathbb{R}^{n \times m}$, and $\mathbf{C}_c \in \mathbb{R}^{n \times 2n}$ are parameters to be learned. Similarly, e_y and e_c can be computed, and consequently α_y and α_c.

4 Experiments

To test our approach, we carry out experiments on the tasks of Chinese-to-English (ZH-EN) and English-to-German (EN-DE) machine translations. All source code is available on github. [1]

4.1 Dataset and Evaluation Metrics

ZH-EN Translation. Our training data for ZH-EN translation consists of 1.25M sentence pairs extracted from LDC corpora, with 27.9M Chinese words and 34.5M English words respectively. NIST MT 06 (1664 sentence pairs) is chosen as the development set while NIST MT 02, 03, 04, 05, and 08 datasets (878, 919, 1788, 1082 and 1357 sentence pairs, respectively) are used as our test sets. We use the case-insensitive 4-gram NIST BLEU score (Papineni et al., 2002) for validation and evaluation, measured by mteval-v11b.pl script.

EN-DE Translation. The EN-DE training data is provided by the standard benchmark ACL WMT 2017 [2], which consists of 5.85M sentence pairs with 141.42M English words and 134.83M German words respectively. We combine news-test-2012 and news-test-2013 as development set (6003 sentence pairs), and use news-test-2014 (News14), news-test-2015 (News15) and news-test-2016 (News16) as test sets (3003, 2169, and 2999 sentence pairs, respectively). Following Barone et al. (2017), we use the validation cross-entropy to choose the best model on the development set and use the case-sensitive 4-gram BLEU score for evaluation on test sets, measured by multi-bleu.perl script.

4.2 Training Details and Systems

We train each model with sentences of length up to 50 words for ZH-EN and 60 words (tokens) for EN-DE. The source and target word embedding dimension is 620. The size of the hidden layer is 1000. We use Adam (Kingma and Ba, 2014) to optimize model parameters with a learning rate of 0.0002, and the mini-batch size of 80.

For efficient training the neural networks, in ZH-EN translation we limit the source and target vocabularies to the most frequent 30K words, covering approximately 97.7% and 99.3% of the data in the two languages respectively. All out of vocabulary words are mapped to a special token *UNK*. For EN-DE translation, we apply byte-pair encoding (BPE) [3] (Sennrich et al., 2016) for better handling unknown words and set the vocabulary size as 30K.

We compare the performance of the following NMT systems:

- baseNMT: We use the open source toolkit dl4mt as our baseline attention-based NMT system (Bahdanau et al., 2015) [4] with most default parameter settings kept the same. For translation, a beam search with size 10 is employed.
- +Adaptive GRU: On baseNMT, we leverage adaptive weighting for GRU, as described in Section 3.2.

[1] https://github.com/liyc7711/weighted-nmt
[2] http://data.statmt.org/wmt17/translation-task/preprocessed/de-en/
[3] https://github.com/rsennrich/subword-nmt
[4] https://github.com/nyu-dl/dl4mt-tutorial

- +Adaptive Output: On baseNMT, we leverage adaptive weighting for the output state, as described in Section 3.3.
- +Both: On baseNMT, we leverage adaptive weighting for both GRU and the output state.

System	MT06	MT02	MT03	MT04	MT05	MT08	All	Params (M)
baseNMT	35.29	39.28	36.69	38.68	36.13	25.69	35.46	89.7
+Adaptive GRU	36.29‡	40.14†	36.99	39.96‡	36.91†	**27.15**‡	36.62‡	97.5
+Adaptive Output	35.72	40.38‡	37.29	39.31†	36.47	25.98	35.93‡	93.1
+Both	**36.52**‡	**40.87**‡	**38.04**‡	**40.41**‡	**37.42**‡	27.07‡	**36.95**‡	100.9

Table 1: BLEU scores of ZH-EN translation. †/‡: significant over baseline at 0.05/ 0.01, tested by bootstrap resampling (Koehn, 2004).

4.3 Experimental Results: ZH-EN Translation

Table 1 shows the performance of ZH-EN translation measured in BLEU score. From the table, we have the following observations.

- NMT models are benefited from adaptive weighting for either GRU or the output state. Moreover, the system of Adaptive GRU outperforms the system of Adaptive Output (i.e., 36.62 vs. 35.93 on all test sets).
- Fortunately, the improvements from adaptive weighting for GRU and the output state have little overlap. Combining the two types of adaptive weighting leads to more improvement on all test sets with the only exception of MT 08. Compared to baseNMT, the system *+Both* yields significant improvement of 1.49 BLEU scores, suggesting the positive effect of dynamically controlling the amount of input information being carried forward.

4.4 Experimental Results: EN-DE Translation

The results on English-German translation are presented in Table 2. It shows that leveraging adaptive weighting for GRU and the output state leads to significant improvement of 0.92 BLEU scores over all test sets. Overall, the performance trend over the proposed systems is consistent with that of ZH-EN translation.

System	News14	News15	News16	All
baseNMT	23.61	25.22	28.79	25.97
+Adaptive GRU	24.26‡	25.71†	29.63‡	26.53 ‡
+Adaptive Output	23.65	25.33	29.34†	26.11
+Both	**24.26**‡	**26.15**‡	**29.98**‡	**26.89**‡

Table 2: BLEU scores of EN-DE translation.

4.5 Parameters and Training Speed

As shown in Table 1, the proposed models introduce new parameters in different ways. [5] The baseline system has 89.8M parameters. The Adaptive GRU system introduces additional 7.8M parameters. The Adaptive output system introduces additional 3.4M parameters.

When running on a single GPU GeForce GTX 1080, the baseline model speeds 1.1 second per batch with 150K updates for ZH-EN, and 600K updates for EN-DE tasks, while the improved system (+Both) only increases the training time by about 12%.

[5]The parameters of the systems for EN-DE translation tasks are same as those for ZH-EN.

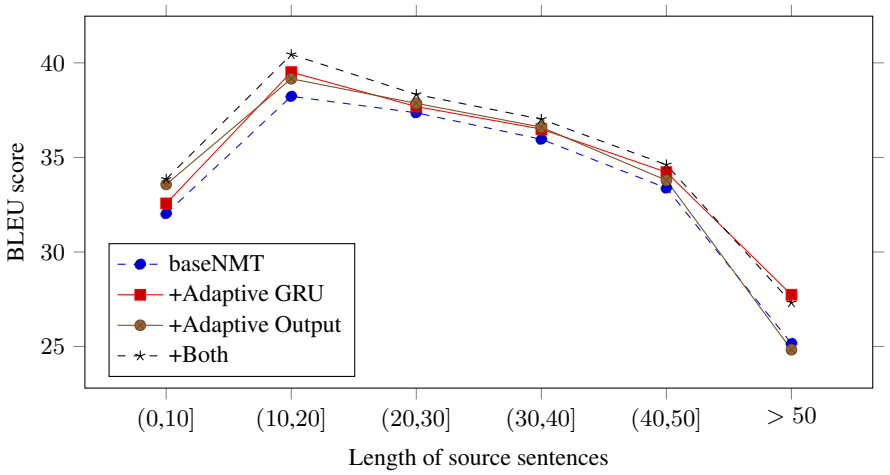

Figure 3: BLEU scores of translation with respect to the lengths of the input sentences.

5 Discussion

In this section, we further look at the *+Both* system and the *baseNMT* system to explore more on how adaptive weighting help in ZH-EN translation.

5.1 Analysis on Word Alignment

In the encoder, the Adaptive GRU uses adaptive weights to control the information flow of previous state and the current input. Thus, it has a direct impact on word representation vectors (i.e., hidden states) of source words. We conjecture that better word representation will help the decoder be able to attend to appropriate source words in decoding. To test this hypothesis, we carry out experiments of the word alignment task on the evaluation dataset from Liu and Sun (2015), which contains 900 manually aligned Chinese-English sentence pairs. We force the decoder to output reference translations, as to get automatic alignments between input sentences and their reference translations. To evaluate alignment performance, we report the alignment error rate (AER) (Och and Ney, 2003) and the soft AER (SAER) (Tu et al., 2016) in Table 3. It shows that adaptive weighting improves the attention model.

System	AER	SAER
baseNMT	43.0	55.7
+Both	**40.9**	**54.6**

Table 3: Evaluation of word alignment for ZH-EN translation. The lower the AER or SAER, the better the alignment quality.

5.2 Effects on Long Sentences

Following Bahdanau et al. (2015), we group sentences of similar lengths together and compute BLEU scores. Figure 3 presents the BLEU scores over different lengths of input sentences. It shows that systems with adaptive weighting consistently outperform baseNMT on all sentence lengths. It also shows that the performance drops substantially when the length of input sentences increases from 20. This performance trend over the length is consistent with the translation output in (Wang et al., 2017; Tu et al., 2016; Li et al., 2017).

5.3 Quantitative Analysis

Due to the continuous representations and non-linearity of neural networks. It is difficult to interpret the internal workings of NMT model (Ding et al., 2017). Fortunately, adaptive weights provide a mechanism to analyze what inputs of a WSM are more important than others. Next, we make insights on what type of

information encoded in the encoder and what types of information has a greater impact on the generation of a target word.

	Chinese POS	**Frequency**	g_t**-F (%)**	g_t**-B (%)**
	All	38,349	41.09	50.20
Content words	verb	8,176	37.84	46.96
	adjective	1,345	38.28	51.50
	adverb	2,104	38.40	50.93
	geographical name	1,537	36.14	50.70
	temporal noun	891	36.85	45.47
	general noun	7,636	39.15	45.57
	person name	587	39.90	44.56
Function words	preposition	1,553	43.59	53.84
	punctuation	4,956	47.91	52.07
	pronoun	1,449	43.01	54.55
	conjunction	1,015	46.27	43.20
	auxiliary	2,891	52.13	57.28

Table 4: Average g_t values groped by part of speech (POS) tags on development set MT06. g_t-F and g_t-B indicate the average g_t values in the forward GRU and the backward GRU, respectively.

Words	α_s **(%)**	α_c **(%)**	α_y**(%)**
All	60.61	29.06	10.34
EOS	72.02	22.40	5.58
.	72.10	20.20	7.7
,	70.00	22.84	7.16
of	66.34	21.34	12.32
to	65.45	22.58	11.96
on	65.72	23.02	11.25
is	71.67	22.32	6.00
has	68.05	26.12	5.84
US	55.42	36.08	8.50
China	56.36	35.01	8.63
president	57.00	34.49	8.52
Kong	42.94	19.30	37.77
York	42.14	26.55	31.31
agency	54.11	11.10	34.79

Table 5: Averaged α_s, α_c and α_y scores in computing the intermediate state o_t for different target words.

Analysis on encoder. In GRU, g_t in Eq. 14 $\sim$ 16 indicates that the importance of h_{t-1}. The lower g_t is, the more important the word content itself x_t and the less important the previous state h_{t-1}. To obtain word representation vectors, the encoder uses a forward GRU and a backward GRU to read an input sentence in two directions. Table 4 presents the average g_t values in the forward GRU and the backward GRU. Interestingly, as shown in the table, the two GRUs behave differently in choosing amount of input word x_t and previous hidden state h_{t-1}. There also exists a consistent trend in the two GRUs that as expected, content words usually have lower g_t than function words, indicating that the encoder focuses more on the words themselves while encoding content words.

Analysis on decoder. In the output states, we assign α_s to control the current hidden state s_t, α_y to the previously generated word y_{t-1} and α_c to the source context c_t. Table 5 lists their average values for a few typical target words in the development set MT06. From it, we observe that:

- Generally speaking, the decoder pays more attention to the current hidden states s_t than either source context c_t or previous target word y_{t-1}.
- As expected, the decoder works differently in selecting how much source context to generate different types of target words. For example, *comma*, *period* and the end of sentence token *EOS* have the lowest α_c scores in that they are mainly decided by the translation content itself and less influenced by source context. This explains why it is surprising that target side *EOS* aligns to source side *EOS* in low frequency (e.g., 20%).
- Similarly, function words, including prepositions (e.g., *of*, *to*, *on*) and auxiliary words (e.g., *is*, *has*) have low α_c scores too, indicating that their generation is less decided by source context.
- Compared to function words, content words (e.g., *US*, *China*, *president*) are opt to have higher α_c scores, suggesting the decoder considers more on source context to generate target side content words.
- However, there are also scenarios that some target content words are less influenced by source context. In 1-to-many translation where a source content word aligns to multiple target content words (i.e., 新华社/Xinhua News Agency, 香港/Hong Kong, and 纽约/New York), the first target content word usually has higher α_c score while the others have lower α_c scores. Figure 4 illustrates the changes of α_s, α_c, and α_y in a real translation *Xinhua News Agency, Hong Kong*. Apart from α_s, it shows a very obvious trend that *Xinhua* and *Hong* have higher α_c scores while *News*, *Agency*, and *Kong* have lower ones.

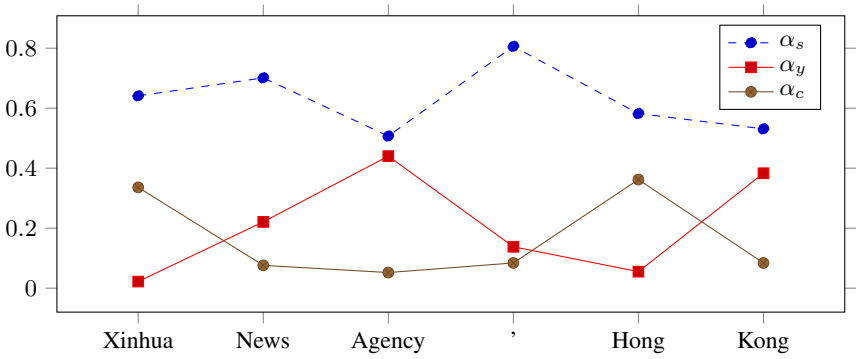

Figure 4: α_s, α_y and α_c scores in a real translation "新华社 香港 / Xinhua News Agency , Hong Kong". In it, there exist two 1-to-many correspondences, i.e., 新华社/Xinhua News Agency, and 香港/Hong Kong.

6 Related Work

We describe related work from two perspectives.

Gate mechanism. Our work is partially inspired by studies of gate mechanism for neural networks. Following the success of LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014a; Cho et al., 2014b), the gate mechanism has become standard components in RNN. Recently, Srivastava et al. (2015) employ gate units to regulate information flow, called highway networks. The most relevant work to ours is Tu et al. (2017), in which they propose context gate to control the ratios of the source context (i.e., c_t) and target context (i.e., y_{t-1} and s_{t-1}) for computing next target state s_t. On the contrary, we use gate units to regulate information flow in computing the output state o_t. Moreover, we propose adaptive weighting for GRU through gate units.

Interpretation for neural networks. Attention mechanism (Bahdanau et al., 2015; Lin et al., 2017; Vaswani et al., 2017) offers a way of understanding the contribution of every source words to the generation of a target word. Ding et al. (2017) propose to use layer-wise relevance propagation (LRP) to

interpret the internal workings of NMT and analyze translation errors. Moreover, Karpathy et al. (2015) and Li et al. (2016) propose to visualize and understand RNNs for natural language processing. In this work, we use the proposed gates in both encoder and decoder to analyze what types of information encoded in the encoder and what types of information influences the generation of a target word.

7 Conclusion

In this paper, we present an approach to regulate the information flow in neural machine translation model explicitly. This is done by employing adaptive weights through gate units. We apply adaptive weighting for both GRU and the output intermediate state. Experimental results on Chinese-to-English and English-to-Germany translation tasks show that the proposed approach achieves better translation performance and alignment quality over baseline NMT system.

Acknowledgements

The authors would like to thank anonymous reviewers for providing helpful comments, and also acknowledge Deyi Xiong, Xing Wang, Mingming Yang, Xiangyu Duan, Zhengxian Gong for useful discussions. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61432013). Junhui and Yachao are also partially supported by the Opening Project of Key Laboratory of China's Ethnic Languages and Information Technology of Ministry of Education (Grant No. KFJJ201605).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*.

Antonio Valerio Miceli Barone, Jindrich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proceedings of WMT 2017*, pages 99–107.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machinetranslation: Encoder-decoder approaches. In *Proceedings of SSST 2014*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, pages 1724–1734.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of ACL 2017*, pages 1150–1159.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of ACL 2017*, pages 123–135.

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

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of WMT 2015*, pages 134–140.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. In *arXiv:1506.02078*.

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

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of NAACL 2016*, pages 681–691.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *Proceedings of ACL 2017*, pages 688–697.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of ICLR 2017*.

Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *Proceedings of AAAI 2015*, pages 857–868.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP 2015*, pages 1412–1421.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*, pages 311–318.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL 2016*, pages 1715–1725.

Rupesh Kumar Srivastava, Klaus Greff, and J ü rgen Schmidhuber. 2015. Highway networks. In *Proceedings of ICML 2015 Deep Learning workshop*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of ACL 2016*, pages 76–85.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS 2017*, pages 6000–6010.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *Proceedings of AAAI 2017*, pages 3330–3336.

Yonghui Wu, Mike Schuster, Zhifeng Chen, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv preprint arXiv:1609.08144*.