Upping the Ante:
Towards a Better Benchmark for Chinese-to-English Machine Translation

Christian Hadiwinoto, Hwee Tou Ng

Department of Computer Science
National University of Singapore
{chrhad, nght}@comp.nus.edu.sg

Abstract

There are many machine translation (MT) papers that propose novel approaches and show improvements over their self-defined baselines. The experimental setting in each paper often differs from one another. As such, it is hard to determine if a proposed approach is really useful and advances the state of the art. Chinese-to-English translation is a common translation direction in MT papers, although there is not one widely accepted experimental setting in Chinese-to-English MT. Our goal in this paper is to propose a benchmark in evaluation setup for Chinese-to-English machine translation, such that the effectiveness of a new proposed MT approach can be directly compared to previous approaches. Towards this end, we also built a highly competitive state-of-the-art MT system trained on a large-scale training set. Our system outperforms reported results on NIST OpenMT test sets in almost all papers published in major conferences and journals in computational linguistics and artificial intelligence in the past 11 years. We argue that a standardized benchmark on data and performance is important for meaningful comparison.

Keywords: machine translation, benchmark, Chinese-to-English

1. Introduction

Over the years, there have been many published papers on machine translation (MT), proposing novel ideas by showing improvements over certain baselines. However, a paper often has a different experimental setup from the others. These variations include the approach (algorithm) and dataset. Ideally, research and development work on MT should be based on a benchmark of system setup with good performance. Otherwise, there is no support in asserting that a proposed method advances the state of the art.

Unfortunately, for Chinese-to-English MT, two widely spoken human languages and one of the most well-studied language translation directions in MT, there is no widely accepted standard benchmark for evaluation, comprising a standardized training set, development set, and test set. Throughout the past decade, Chinese-to-English translation has been most commonly performed on NIST OpenMT[1] test sets, trained on parallel and monolingual corpora from the Linguistic Data Consortium (LDC)[2].

Our goal in this paper is to propose a benchmark in evaluation setup for Chinese-to-English machine translation, such that the effectiveness of a new proposed MT approach can be directly compared to previous approaches. Towards this end, we also built a highly competitive state-of-the-art MT system trained on a large-scale training set. Our system outperforms reported results on NIST OpenMT test sets in almost all papers published in major conferences and journals in computational linguistics and artificial intelligence in the past 11 years.

The rest of this paper is organized as follows. Section 2 describes our MT approach. Section 3 elaborates our experimental setup. Section 4 presents our experimental results. Section 5 describes related work. Finally, Section 6 gives the conclusion.

2. Neural Machine Translation

We built a neural machine translation (NMT) system based on the encoder-decoder approach with attention mechanism (Bahdanau et al., 2015). This NMT approach encodes an input sentence into a continuous representation by an encoder recurrent neural network (RNN) and produces translation output by a decoder RNN. The decoder RNN, through an attention mechanism, looks into different parts of the encoded input sentence while decoding is in progress.

2.1. Encoder-Decoder Model with Attention

Given a target language sentence $\mathbf{y} = (y_1, ..., y_n)$ and the corresponding source language sentence $\mathbf{x} = (x_1, ..., x_m)$, the neural machine translation model is formulated as

$$p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) \tag{1}$$

in which the probability of the target word y_i at time step i is computed by the decoder RNN as follows:

$$p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = F(y_i, y_{i-1}, s_i, c_i) = t_i[y_i] \tag{2}$$

where F is a function to compute the probability of the word y_i to be generated at time step i and t_i is a vector having the size of the target language vocabulary, in which each vector dimension $t_i[y]$ stores the probability of a word y, computed as follows:

$$t_i = \mathrm{softmax}(W_t(\tanh(U_t s_i + V_t E[y_{i-1}] + C_t c_i + b_t))) \tag{3}$$

where U_t, V_t, and C_t are matrices mapping the hidden state s_i, the embedding of the previous word $E[y_{i-1}]$, and the context vector c_i respectively to an intermediate vector representation, with b_t being the bias vector. Then

W_t transforms the intermediate vector representation to a vocabulary-sized probability vector.

The decoder hidden state at a time step i is computed by

$$s_i = g_y(E[y_{i-1}], s_{i-1}, c_i) \qquad (4)$$

where g_y is the RNN unit function to compute the current hidden state given the hidden state of the previous time step, the previous word embedding, and the context.

Equation 3 indicates that the target word to be generated at a given time step takes into account the context vector c_i, which is a weighted sum of each annotation vector h_j, representing the source language sentence at position j:

$$c_i = \sum_{j=1}^{m} \alpha_{ij} h_j \qquad (5)$$

in which the scalar weight α_{ij} for each h_j is computed by a softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{m} \exp(e_{ik})} \qquad (6)$$

where e_{ij} is computed by

$$e_{ij} = v_a^T \tanh(W_a s_i' + U_a h_j + b_a) + \beta \qquad (7)$$

where W_a and U_a are the weight matrices and b_a is the bias vector to compute a vector, which is then converted by the weight vector v_a and the bias term β into a scalar e_{ij}, i.e., the degree of matching between the target word at time step i and the input word at position j. This is conceptually a soft alignment model.

To compute the decoding hidden state s_i in Equation 4, we adopt an approach that incorporates the context c_i from the attention mechanism by using two transitions (Sennrich et al., 2017b). The decoder hidden state function $g_y(E[y_{i-1}], s_{i-1}, c_i)$ in Equation 4 first passes the embedding $E[y_{i-1}]$ of the input word y_{i-1} to the first recurrent unit function, resulting in an intermediate hidden state s_i', which is computed by the decoder recurrent unit functions $g_{y,1}$ as:

$$s_i' = g_{y,1}(E[y_{i-1}], s_{i-1}) \qquad (8)$$

and is passed to Equation 7. Then, the second recurrent unit function $g_{y,2}$ processes the context c_i defined in Equation 5 and the intermediate hidden state s_i' as follows:

$$s_i = g_{y,2}(c_i, s_i') = g_{att}(E[y_{i-1}], s_{i-1}, c_i) \qquad (9)$$

where $g_{att}(E[y_{i-1}], s_{i-1}, c_i)$ is a composition of $g_{y,1}$ and $g_{y,2}$. It is to be noted that as the decoder generates an output word y_i at current time step i, the input word at the time step is y_{i-1}. The recurrent unit function is described further in Section 2.2.

We made use of the bidirectional encoder RNN, where each annotation vector h_j is a concatenation of the forward and the backward RNN hidden states, $\overrightarrow{h_j}$ and $\overleftarrow{h_j}$ respectively, defined as follows:

$$h_j = [\overrightarrow{h}_j; \overleftarrow{h}_j] \qquad (10)$$

$$\overrightarrow{h}_j = \overrightarrow{g}_x(E[x_j], \overrightarrow{h}_{j-1}) \qquad (11)$$

$$\overleftarrow{h}_j = \overleftarrow{g}_x(E[x_j], \overleftarrow{h}_{j+1}) \qquad (12)$$

where $\overrightarrow{g}_x$ is the forward RNN unit function to compute the RNN hidden state at the current encoding position j given the embedding of the current word $E[x_j]$ and the hidden state at the previous position, while $\overleftarrow{g}_x$ is the backward RNN unit function to compute the hidden state at j given the word embedding $E[x_j]$ and the hidden state at the next position.

Training an end-to-end NMT model is conducted by the back-propagation through time (BPTT) algorithm, which updates the parameters of the RNN while the time steps are unrolled, to minimize a cost function. The parallel training corpus is divided into mini-batches, each consisting of N parallel sentences. The NMT model parameters are updated in each mini-batch.

Translation decoding is performed by a beam search algorithm, which produces translation output sequentially in the target language order. NMT decoding proceeds by generating one word at each time step.

In NMT, as described in Equation 3, computing the probability involves mapping the hidden state vector to a vector with the dimension of the vocabulary size. Therefore, to make computation tractable, the NMT vocabulary size is limited. To cope with the limitation of the vocabulary size, we adopt fragmentation of words into sub-words of character sequences through the byte pair encoding (BPE) algorithm (Sennrich et al., 2016). This algorithm finds the N most frequent character sequences of variable length, through N character merge operations, and splits less frequent words based on this list of character sub-sequences.

2.2. Recurrent Unit Function

To compute the hidden state representations in Equations 4 and 10–12, we made use of recurrent unit functions with gate mechanism to control the flow of information from the input and the previous hidden state. There are two commonly adopted gate mechanisms in the encoder-decoder RNN NMT model, namely the long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (GRU) (Cho et al., 2014).

The LSTM RNN unit consists of a memory cell μ_j and three gates, i.e., the input gate ι_j that controls the intensity of the new information to be stored in the memory cell, the forget gate f_j that controls how much to remember or to forget from the previous memory cell, and the output gate o_j that controls how much information is output to the hidden state from the memory. At each time step j, given the input χ_j, the hidden state η_j is formulated as:

$$\begin{aligned} \eta_j &= LSTM(\chi_j, \eta_{j-1}) \\ &= o_j \circ \tanh(\mu_j) \end{aligned} \qquad (13)$$

where

$$\iota_j = \sigma(W_\iota \chi_j + U_\iota \eta_{j-1} + b_\iota)$$
$$f_j = \sigma(W_f \chi_j + U_f \eta_{j-1} + b_f)$$
$$\mu_j = f_j \circ \mu_{j-1} + \iota_j \circ \tanh(W_\mu \chi_j + U_\mu \eta_{j-1} + b_\mu)$$
$$o_j = \sigma(W_o \chi_j + U_o \eta_{j-1} + b_o)$$

W and U denote the weight matrices transforming the input embedding and the previous hidden state into the corresponding outputs, and b denotes the bias vectors.

Meanwhile, the recurrent unit for GRU at each time step j consists of two gates, i.e., the update gate z_j and the reset gate r_j. At each time step j, given the time step input χ_j, the hidden state η_j is formulated as:

$$
\begin{aligned}
\eta_j &= GRU(\chi_j, \eta_{j-1}) \\
&= (1 - z_j) \circ \eta_{j-1} + z_j \circ \underline{\eta}_j
\end{aligned}
\tag{14}
$$

where

$$
\begin{aligned}
\underline{\eta}_j &= \tanh(W\chi_j + U[r_j \circ \eta_{j-1}] + b) \\
z_j &= \sigma(W_z \chi_j + U_z \eta_{j-1} + b_z) \\
r_j &= \sigma(W_r \chi_j + U_r \eta_{j-1} + b_r)
\end{aligned}
$$

W and U denote the weight matrices transforming the input embedding and the previous hidden state to the corresponding outputs (denoted by the subscript), and b denotes the bias vectors.

As shown in Equations 13 and 14, both LSTM and GRU make use of gate mechanisms to control the information flow from the input and the hidden state. But unlike LSTM, GRU does not have the memory cell and the output gate. GRU proposes the hidden state $\underline{\eta}_j$ and interpolates each dimension with that of the previous hidden state, controlled by the update gate z_j. Meanwhile, the reset gate r_j controls the intensity of the previous hidden state to be taken into account in the current pre-computed hidden state.

The LSTM encoder re-defines Equations 11 and 12 respectively as:

$$
\begin{aligned}
\overrightarrow{h}_j &= \overrightarrow{g}_x(E[x_j], \overrightarrow{h}_{j-1}) = \overrightarrow{LSTM}(E[x_j], \overrightarrow{h}_{j-1}) \\
\overleftarrow{h}_j &= \overleftarrow{g}_x(E[x_j], \overleftarrow{h}_{j+1}) = \overleftarrow{LSTM}(E[x_j], \overleftarrow{h}_{j+1})
\end{aligned}
$$

while the GRU encoder re-defines the two equations respectively as:

$$
\begin{aligned}
\overrightarrow{h}_j &= \overrightarrow{g}_x(E[x_j], \overrightarrow{h}_{j-1}) = \overrightarrow{GRU}(E[x_j], \overrightarrow{h}_{j-1}) \\
\overleftarrow{h}_j &= \overleftarrow{g}_x(E[x_j], \overleftarrow{h}_{j+1}) = \overleftarrow{GRU}(E[x_j], \overleftarrow{h}_{j+1})
\end{aligned}
$$

with the embedding representation $E[x_j]$ of word x_j at position j as the input to the recurrent unit function.

For decoding, in Equations 8 and 9, both $g_{y,1}$ and $g_{y,2}$ can be instantiated by LSTM (Equation 13) as:

$$
\begin{aligned}
g_{y,1}(E[y_{i-1}], s_{i-1}) &= LSTM_1(E[y_{i-1}], s_{i-1}) \\
g_{y,2}(c_i, s'_i) &= LSTM_2(c_i, s'_i)
\end{aligned}
$$

or with GRU (Equation 14) as:

$$
\begin{aligned}
g_{y,1}(E[y_{i-1}], s_{i-1}) &= GRU_1(E[y_{i-1}], s_{i-1}) \\
g_{y,2}(c_i, s'_i) &= GRU_2(c_i, s'_i)
\end{aligned}
$$

where the input to the LSTM or GRU function for $g_{y,1}$ is the embedding of the previous word $E[y_{i-1}]$ and the input for $g_{y,2}$ is the context vector c_i.

2.3. Deep Recurrent Layers

Following (Sennrich et al., 2017a), we adopt deep RNN models for encoding and decoding. There are two alternatives to achieve this, namely the deep stacked RNN and

deep transition RNN. The deep stacked RNN passes the whole input sequence to the first layer of RNN and feeds the sequence of the output hidden representations to the next layer of RNN. This is done subsequently depending on the number of RNN layers. Meanwhile, the deep transition RNN passes an input at each time step through a series of transitions (i.e., recurrent unit functions) and passes the hidden state of the last transition to the next time step.

2.3.1. Deep Stacked RNN

We adopt the deep stacked RNN that involves residual connection, summing the output of the previous RNN layer with the computed hidden state of the current RNN layer, and alternation of direction (Zhou et al., 2016), as illustrated in Figure 1. In this model, for each layer l and time step j, we need to distinguish between the computed hidden state of the current RNN unit function without residual connection, i.e., $\overrightarrow{h}_j^l$, and the hidden state which includes the residual connection, i.e., $\overrightarrow{w}_j^l$. The alternation of direction is designed such that the odd-numbered and even-numbered RNN layers process the sequence in the left-to-right and right-to-left directions respectively. In the stacked RNN with layer depth D_x, the forward encoder hidden state at time step j in Equation 11, $\overrightarrow{h}_j$, is computed as follows:

$$
\begin{aligned}
\overrightarrow{h}_j &= \overrightarrow{g}_x(E[x_j], \overrightarrow{h}_{j-1}) = \overrightarrow{w}_j^{D_x} \\
\overrightarrow{w}_j^{D_x} &= \overrightarrow{h}_j^{D_x} + \overrightarrow{w}_j^{D_x - 1} \\
\overrightarrow{w}_j^1 &= \overrightarrow{h}_j^1 = \overrightarrow{g}_x^1(E[x_j], \overrightarrow{h}_{j-1}^1) \\
\overrightarrow{h}_j^{2k} &= \overrightarrow{g}_x^{2k}(\overrightarrow{w}_j^{2k-1}, \overrightarrow{h}_{j+1}^{2k}), \text{ for } 1 < 2k \leq D_x \\
\overrightarrow{h}_j^{2k+1} &= \overrightarrow{g}_x^{2k+1}(\overrightarrow{w}_j^{2k}, \overrightarrow{h}_{j-1}^{2k+1}), \text{ for } 1 < 2k+1 \leq D_x \\
\overrightarrow{w}_j^l &= \overrightarrow{h}_j^l + \overrightarrow{w}_j^{l-1}, \text{ for } 1 < l \leq D_x
\end{aligned}
$$

In the above equations, $\overrightarrow{g}_x^l$ is the forward encoder recurrent unit function of a layer l in the deep RNN stack. It can be instantiated with LSTM or GRU. While the above equations compute the forward encoder hidden state, the backward encoder hidden state $\overleftarrow{h}_j$ is computed similarly by changing the arrow direction from right ($\rightarrow$) to left ($\leftarrow$) and swapping $j - 1$ with $j + 1$.

Since at each time step, the decoder has no knowledge of the next word, there is no alternation of direction therein. In addition, we only use the recurrent function with attention in the first layer of the RNN stack s_i^1, while the deeper layers are simple RNN without attention. Therefore, the decoder RNN with layer depth D_y computes the hidden state s_i in Equation 4, as follows:

$$
\begin{aligned}
s_i &= g_y(E[y_{i-1}], s_{i-1}, c_i) = w_i^{D_y} \\
w_i^{D_y} &= s_i^{D_y} + w_i^{D_y - 1} \\
s_i^{D_y} &= g_y^{D_y}(w_i^{D_y - 1}, s_{i-1}^{D_y}) \\
w_i^1 &= s_i^1 = g_{att}^1(E[y_{i-1}], s_{i-1}^1, c_i) \\
s_i^l &= g_y^l(w_i^{l-1}, s_{i-1}^l), \text{ for } 1 < l \leq D_y \\
w_i^l &= s_i^l + w_i^{l-1}, \text{ for } 1 < l \leq D_y
\end{aligned}
$$

The first RNN stack layer, $s_i^1 = g_{att}^1(E[y_{i-1}], s_{i-1}^1, c_i)$, is a composition of two recurrent unit functions like in Equa-
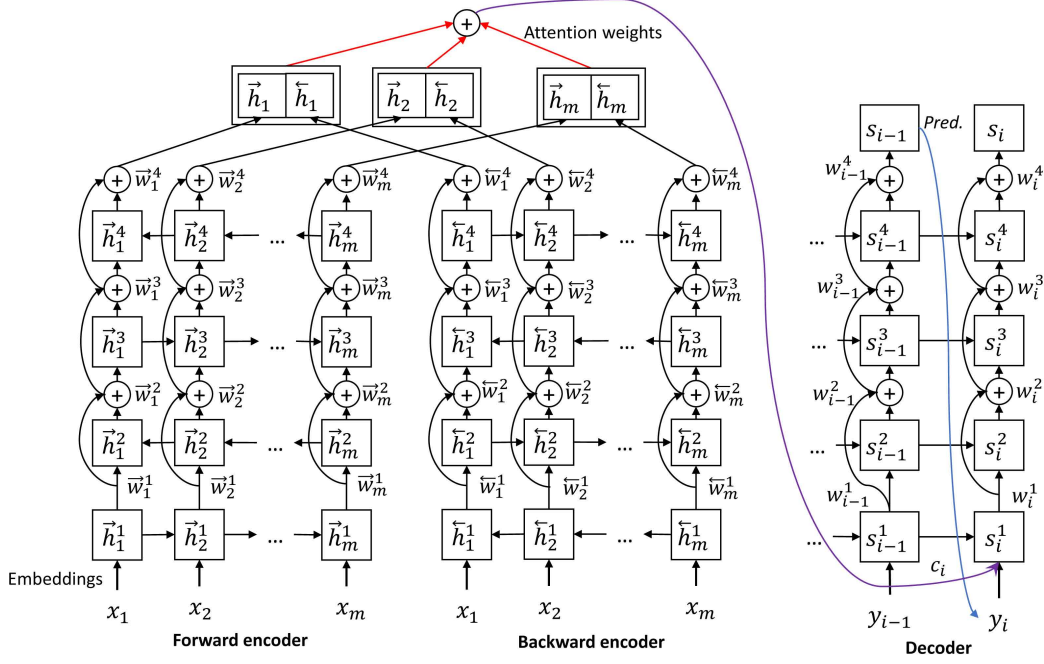
Figure 1: An illustration of a deep stacked RNN model (Zhou et al., 2016) with encoder stack depth (D_x) of 4 and decoder stack depth (D_y) of 4.

tions 8 and 9, i.e.,

$$s_i^1 = g_{att}^1(E[y_{i-1}], s_{i-1}^1, c_i)$$
$$s_i' = g_{y,1}^1(E[y_{i-1}], s_{i-1}^1)$$
$$s_i^1 = g_{y,2}^1(c_i, s_i')$$

Like the encoders, the decoder recurrent unit function g_y^l at each layer l can be instantiated by LSTM or GRU.

2.3.2. Deep Transition RNN

The deep transition RNN (Miceli-Barone et al., 2017) involves a number of layers within a time step j through which an input word is fed, as illustrated in Figure 2. The recurrent unit function of each layer l is defined as a transition, which outputs an intermediate state $\overrightarrow{h}_{j,l}$ for the encoder and $s_{j,l}$ for the decoder. With L_x transitions, the hidden state representation at j is equivalent to the output of the last layer, so for the forward encoder state, Equation 11 defines $\overrightarrow{h}_j$ as:

$$\overrightarrow{h}_j = \overrightarrow{g}_x(E[x_j], \overrightarrow{h}_{j-1}) = \overrightarrow{h}_{j,L_x}$$
$$\overrightarrow{h}_{j,1} = \overrightarrow{g}_{x,1}(E[x_j], \overrightarrow{h}_{j-1,L_x})$$
$$\overrightarrow{h}_{j,l} = \overrightarrow{g}_{x,l}(0, \overrightarrow{h}_{j,l-1}), \text{ for } 1 < l \leq L_x$$

where the input to the first recurrent unit transition is $E[x_j]$, the embedding of the input word at j, while the subsequent higher level transitions only receive the output from the previous transition and do not receive the input word. The reverse hidden state $\overleftarrow{h}_j$ in Equation 12 is computed similarly by substituting $j - 1$ with $j + 1$. The encoder recurrent unit function $\overrightarrow{g}_{x,l}$ at each layer l can be instantiated by LSTM or GRU.

While the baseline shallow decoder RNN already contains two transitions, without and with attention respectively, the

deep transition decoder RNN is extended similarly to the deep transition encoder RNN, such that the decoder hidden state with depth L_y is computed as:

$$s_{i,1} = s_i' = g_{y,1}(E[y_{i-1}], s_{i-1,L_y})$$
$$s_{i,2} = g_{y,2}(c_i, s_{i,1})$$
$$s_{i,l} = g_{y,l}(0, s_{i,l-1}), \text{ for } 2 < l \leq L_y$$
$$s_i = s_{i,L_y}$$

Similarly, the decoder recurrent unit function $g_{y,l}$ at each layer l can be instantiated by LSTM or GRU.

3. Experimental Setup

3.1. Datasets

We conducted experiments using the parallel training corpora from LDC to test on the NIST test sets. In addition, we also conducted experiments on the United Nations Parallel Corpus (Ziemski et al., 2016), following (Junczys-Dowmunt et al., 2016). We used the pre-defined training, development, and test sets of the corpus following (Junczys-Dowmunt et al., 2016) and conducted NMT experiments accordingly.

We pre-processed our parallel training corpora by segmenting Chinese sentences, which originally have no spaces to demarcate words, and tokenizing English sentences to split punctuation symbols from words. Chinese word segmentation was performed by a maximum entropy model (Low et al., 2005) trained on the Chinese Penn Treebank (CTB) segmentation standard.

To alleviate the effect of rare words in NMT, we fragmented words to sub-words through the byte pair encoding (BPE) algorithm (Sennrich et al., 2016) with 59,500 merge operations. All our training sentences are lowercased.
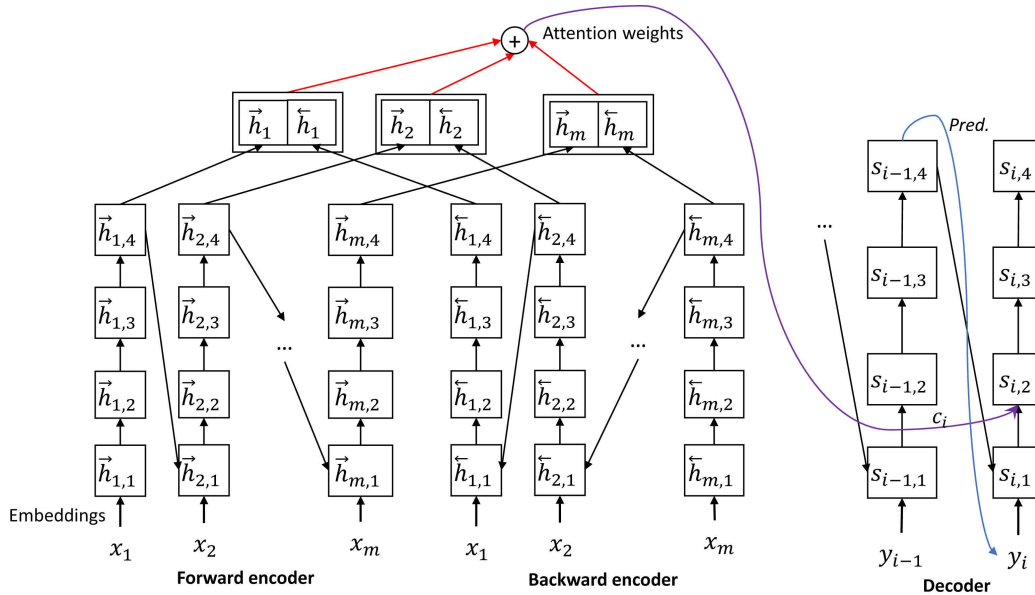
19

Figure 2: An illustration of a deep transition RNN model (Miceli-Barone et al., 2017) with 4 encoder transitions ($L_x = 4$) and 4 decoder transitions ($L_y = 4$).

3.1.1. LDC Corpora

We divide the LDC corpora we used into older corpora[3] and newer corpora[4]. Due to the dominant older corpora, we duplicate the newer corpora of various domains ten times to achieve better domain balance.

In addition to the parallel sentences, we also utilized a large amount of monolingual English texts, consisting of the English side of FBIS parallel corpus (LDC2003E14) and all the sub-corpora of the English Gigaword Fourth Edition (LDC2009T07). Altogether, the combined corpus consists of 107M sentences and 3.8B tokens. Each individual Gigaword sub-corpus (i.e., AFP, APW, CNA, LTW, NYT, and Xinhua) is used to train a separate N-gram language model. The English side of FBIS is also used to train another separate language model (LM). These individual language models are then interpolated to build one single large LM, via perplexity tuning on the English side of the development data. We use this LM for translation output re-ranking.

To recover the original casing on the translation output, we trained a statistical MT recaser model by using Moses (Koehn et al., 2007) on the English side of FBIS parallel text and the Xinhua portion of English Gigaword Fourth Edition.

Due to computation time and memory consideration, parallel sentences in the corpora that are longer than 50 sub-words in either Chinese or English are discarded. In the end, the final parallel training examples consist of 7.65M sentence pairs, 169M Chinese sub-word tokens (equivalent to 166M word tokens), and 186M English sub-word tokens (equivalent to 184M word tokens).

Our translation development (tuning) set is MTC corpus version 1 (LDC2002T01) and version 3 (LDC2004T07). This development set has 1,928 sentence pairs in total, 49K Chinese word tokens and 58K English word tokens on average across the four reference translations. Our translation test set consists of the NIST MT evaluation sets from 2002 to 2006, and 2008[5]. Altogether in the test sets, there are 7,497 sentence pairs, 192K Chinese word tokens, and 237K English word tokens on average across the four reference translations.

3.1.2. UN Parallel Corpus

The training set of the UN Parallel Corpus, after pre-processing and filtering those exceeding 50 sub-words, consists of 9.73M parallel sentence pairs, 207M Chinese sub-word tokens (equivalent to 204M word tokens), and 225 English sub-word tokens (equivalent to 223M word tokens). We also utilized larger English monolingual text, i.e., all the English side of the UN Parallel Corpus before length filtering, consisting of 11.3M sentences and 335M word tokens. The development set of the UN Parallel Corpus contains 4,000 sentence pairs, 107K Chinese word tokens, and 118K English word tokens, while the test set contains 4,000 sentence pairs, 106K Chinese word tokens, and 118K English word tokens. There is only one reference translation in the development and test sets of the UN Parallel Corpus.

3.2. NMT Model Parameters

We built our neural machine translation (NMT) system by using Nematus (Sennrich et al., 2017b), an open-source NMT toolkit which implements the encoder-decoder NMT architecture with attention mechanism. Our system is based on the NMT system in (Sennrich et al., 2017a). We built an

---

[3] LDC2002E18, LDC2003E14, LDC2004E12, LDC2004T08, LDC2005T06, and LDC2005T10.

[4] LDC2007T23, LDC2008T06, LDC2008T08, LDC2008T18, LDC2009T02, LDC2009T06, LDC2009T15, LDC2010T03, LDC2013T11, LDC2013T16, LDC2014T04, LDC2014T11, LDC2014T15, LDC2014T20, and LDC2014T26.

[5] LDC2010T10, LDC2010T11, LDC2010T12, LDC2010T14, LDC2010T17, and LDC2010T21.

ensemble model consisting of 4 independent models, which are the cross product of two different deep RNN architectures, i.e., deep stacked RNN and deep transition RNN, and two different recurrent unit functions, i.e., GRU and LSTM. For all our models, the word embedding dimension is 500, and the hidden layer dimension is 1,024. Our deep stacked RNN contains 4 stack layers on each of the encoder and decoder. Meanwhile, our deep transition RNN contains 4 encoder transitions and 8 decoder transitions.

Training for each individual model progresses by updating the model parameters at each mini-batch of 40 sentence pairs to minimize the negative log-likelihood loss function on the parallel training data. We use the Adam algorithm (Kingma and Ba, 2015) with learning rate of 0.0001. At each update, we clip the gradient norm to 1.0. We apply layer normalization (Ba et al., 2016) on the model parameters for faster convergence and tie the target-side embedding with the transpose of the output weight matrix (Press and Wolf, 2017). Model parameters are saved at every checkpoint of 10,000 update iterations. At this stage, the negative log-likelihood loss function on the development set is checked. Training stops when there has been no improvement over the lowest loss function value on the development set for 10 consecutive checkpoints.

The main difference between our system and (Sennrich et al., 2017a) is that while they only built NMT models with GRU, we also made use of LSTM. Another difference is in the usage of the larger monolingual English text. They built a synthetic Chinese-English parallel corpus by translating the monolingual English text to Chinese with an English-to-Chinese (reverse direction) NMT model and appended those sentence pairs to their parallel training corpus. Meanwhile, we exploited the English monolingual corpus by building a 5-gram language model to re-rank the k-best translation outputs produced by our NMT system.

4. Experimental Results

For experiments using the LDC corpora, translation quality is measured by case-insensitive BLEU (Papineni et al., 2002), for which the brevity penalty is computed based on the shortest reference (NIST-BLEU)[6]. Statistical significance testing between systems is conducted by bootstrap resampling (Koehn, 2004).

As shown in Table 1, among the individual model types, the deep stacked LSTM NMT model gives the best performance. However the best result is achieved by the ensemble of 4 independent model types, combining deep stacked and deep transition architectures with GRU and LSTM recurrent unit functions. This ensemble model gives an improvement of 4.40 BLEU points over the best deep stacked LSTM model. Applying re-ranking by the N-gram language model on top of the ensemble system of 4 independent models gives a further improvement of 0.11 BLEU point on average, which gives the best result of our system. We are interested in comparing our 4-model ensemble before re-ranking with the ensemble of 2 models with only one RNN unit function (but two different deep layers) and

with only one deep layer (but two different RNN units). As shown in Table 2, our 4-model ensemble outperforms every 2-model ensemble, and the improvement is statistically significant ($p < 0.01$).

We also compare the best results of our system on NIST test sets with the best results reported in published papers on Chinese-to-English MT systems in major computational linguistics and artificial intelligence publication venues[7]. Over the years 2007–2017 (both years inclusive), our 4-model ensemble system with re-ranking achieves a higher BLEU score than the best results reported in almost all (402 out of 403) papers[8].

Note that the LDC training datasets used in the published papers that we compare to may not be the same as ours. This incomparability would not have happened if there were a widely adopted, standardized dataset for training.

In addition, since there are two ways of computing BLEU scores with respect to word casing, i.e., case-insensitive and case-sensitive, we have taken care to compare BLEU scores using the same word casing, that is, by comparing our case-sensitive BLEU scores only to case-sensitive BLEU scores published previously, and similarly for case-insensitive BLEU scores.

Moreover, there are two ways of computation with respect to brevity penalty calculation involving multiple reference translations, namely "shortest", used in NIST-BLEU, and "closest", used in IBM-BLEU. The former sets brevity penalty against the shortest reference translation while the latter sets brevity penalty against the reference translation whose length is the most similar to the system translation output. We have also taken this into account by comparing NIST-BLEU with NIST-BLEU, and IBM-BLEU with IBM-BLEU[9].

For UN Parallel Corpus experiments, translation quality is measured by case-insensitive BLEU using the script provided by Moses[10], following the evaluation setup in (Junczys-Dowmunt et al., 2016)[11].

As shown in Table 3, our best result is obtained by an ensemble of 4 independent models with k-best output re-ranking using N-gram LM trained on the whole English side of the UN Parallel Corpus. Our system with re-ranking is 0.3 BLEU point better than without re-ranking, and the improvement is statistically significant ($p < 0.05$). Both of our systems achieve higher BLEU scores than the best published result for Chinese-to-English translation reported in (Junczys-Dowmunt et al., 2016).

---

[7]CL journal, TACL, ACL, COLING, EMNLP, NAACL, SSST, WMT, AAAI, and IJCAI.

[8]In one paper out of 403 papers (Huang et al., 2013), testing was performed on 4 test subsets: the news subset and the web subset of NIST06 and NIST08. Our system is only marginally worse (by less than 0.1 BLEU point) on the news subset of NIST06, and is better by a large margin (by 2–6 BLEU points) on the other 3 subsets.

[9]ftp://jaguar.ncsl.nist.gov/mt/resources/ mteval-v13a.pl

[10]The multi-bleu.perl script in the Moses distribution.

[11]Personal communication.

---

[6]ftp://jaguar.ncsl.nist.gov/mt/resources/ mteval-v11b.pl

Dataset	GRU		LSTM		4mod-ens	
	dstack	dtrans	dstack	dtrans	no re-ranking	with re-ranking
NIST02	41.27	43.28	44.03	42.12	46.82**	46.94
NIST03	41.78	42.38	42.49	41.87	47.42**	47.58
NIST04	43.75	44.33	45.11	43.97	49.12**	49.13
NIST05	41.71	42.52	43.40	41.97	47.72**	47.78
NIST06	42.27	43.18	43.43	42.19	49.19**	49.37
NIST08	35.28	36.11	36.78	35.55	41.36**	41.48
Average	41.01	41.97	**42.54**	41.28	**46.94****	**47.05**

Table 1: Experimental results in BLEU (%) of our NMT systems on NIST data set from LDC. Each individual model is obtained by cross-combining two different deep RNN architectures, i.e., deep stacked (**dstack**) and deep transition (**dtrans**) RNN, with two different recurrent unit functions, i.e., **GRU** and **LSTM**, without k-best re-ranking. The ensemble of 4 model types (**4mod-ens**) is obtained by taking the best model from each individual model type. This setting is tested both without and with re-ranking. Statistical significance testing was done to compare **4mod-ens** with the best individual model type, **dstack-LSTM** (**: significant at $p < 0.01$).

Layer / Unit	dstack	dtrans	dstack +dtrans
GRU	41.01	41.97	45.15
LSTM	42.54	41.28	45.34
GRU+LSTM	44.72	45.02	**46.94****††‡‡##

Table 2: Experimental results in BLEU (%) on NIST data set showing different combinations of RNN **unit** functions and deep **layers**. The RNN unit functions include GRU, LSTM, and the ensemble of the two. The deep layers include deep stacked (**dstack**) and deep transition (**dtrans**) RNN, and the ensemble of the two. Statistical significance testing is shown to compare our 4-model ensemble with the ensemble of **dstack+dtrans GRU** (**: significant at $p < 0.01$), with **dstack+dtrans LSTM** (††: significant at $p < 0.01$), with **dstack GRU+LSTM** (‡‡: significant at $p < 0.01$), and with **dtrans GRU+LSTM** (##: significant at $p < 0.01$).

Published	Ours (4mod-ens)	
	no reranking	with reranking
53.1	55.0	55.3*

Table 3: Experimental results in BLEU (%) on the test set of the UN Parallel Corpus of the best published result in (Junczys-Dowmunt et al., 2016) and our system, without and with k-best re-ranking with N-gram language model. Statistical significance testing shows the comparison between our 4-model system with re-ranking and without re-ranking (*: significant at $p < 0.05$).

5. Related Work

Establishing standards for the state of the art by publicly accessible resources is important in research. In speech recognition, for instance, there has been work on building a virtual machine as a means of collaboratively building a state-of-the-art system for speech recognition (Metze et al., 2013), aiming at realizing a standardized state-of-the-art system in a collaborative manner. While we do not provide any virtual machines, we have a similar intention of making available a state-of-the-art MT system.

On NMT, Denkowski and Neubig (2017) argued that experiments should be based on a strong baseline system to ensure that a newly proposed approach indeed improves over the best prior published approaches. They performed their experiments on WMT and IWSLT tasks (but not on Chinese-to-English translation) which have fixed training, development, and test sets. The problem for Chinese-to-English MT is greater in that there is no pre-defined set of training data that must be used for experiments, and various groups used different tuning sets and reported their results on different NIST OpenMT test sets. In addition, the lack of a standardized benchmark is not limited to neural MT approaches, but has been widespread since statistical MT approaches began to be tested on Chinese-to-English translation.

6. Conclusion

The problem of lack of consistent experimental setups for Chinese-to-English MT poses a challenge in evaluating newly proposed approaches over the pre-existing state of the art. This can be avoided if there is a clear benchmark which consists of standardized training, development, and test sets, as well as a common benchmark system setup. In this paper, we have shown that our proposed MT approach can be used to build a competitive system on the NIST OpenMT test sets that outperforms systems in almost all 403 published papers in the past 11 years.

We encourage Chinese-to-English MT experiments to use our common benchmark consisting of standard data and evaluation. As the NIST dataset from LDC is widely used in the Chinese-to-English MT research community, we have put up a scoreboard listing the scores achieved by all prior published papers when evaluated on the NIST dataset and released the source code and translation output of our best NMT system[12]. By doing so, we hope future work can make more meaningful comparisons to previous MT research.

---

[12] https://github.com/nusnlp/
c2e-mt-benchmark

7. Bibliographical References

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *CoRR*, abs/1607.06450.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Denkowski, M. and Neubig, G. (2017). Stronger baselines for trustable results in neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 18–27.

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

Huang, Z., Devlin, J., and Zbib, R. (2013). Factored soft source syntactic constraints for hierarchical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 556–566.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? A case study on 30 translation directions. In *Proceedings of the 13th International Workshop on Spoken Language Translation*.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Low, J. K., Ng, H. T., and Guo, W. (2005). A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 161–164.

Metze, F., Fosler-Lussier, E., and Bates, R. (2013). The speech recognition virtual kitchen. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (Interspeech-2013)*, pages 1858–1860.

Miceli-Barone, A. V., Helcl, J., Sennrich, R., Haddow, B., and Birch, A. (2017). Deep architectures for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 99–107.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Press, O. and Wolf, L. (2017). Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*, pages 157–163.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Miceli-Barone, A. V., and Williams, P. (2017a). The University of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Conference on Machine Translation: Shared Task Papers*, pages 389–399.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Miceli-Barone, A. V., Mokry, J., and Nădejde, M. (2017b). Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68.

Zhou, J., Cao, Y., Wang, X., Li, P., and Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4(1):371–383.

Ziemski, M., Junczys-Dowmunt, M., and Pouliquen, B. (2016). The United Nations Parallel Corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.