Empirical Study of Dropout Scheme for Neural Machine Translation

Xiaolin Wang xiaolin.wang@nict.go.jp
Masao Utiyama mutiyama@nict.go.jp
Eiichiro Sumita eiichiro.sumita@nict.go.jp
Advanced Translation Research and Development Promotion Center,
National Institute of Information and Communications Technology, Japan

Abstract

Dropout has lately been recognized as an effective method to relieve over-fitting when training deep neural networks. However, there has been little work studying the optimal dropout scheme for neural machine translation (NMT). NMT models usually contain attention mechanisms and multiple recurrent layers, thus applying dropout becomes a non-trivial task. This paper approached this problem empirically through experiments where dropout were applied to different parts of connections using different dropout rates. The work in this paper not only leads to an improvement over an established baseline, but also provides useful heuristics about using dropout effectively to train NMT models. These heuristics include which part of connections in NMT models have higher priority for dropout than the others, and how to correctly enhance the effect of dropout for difficult translation tasks.

1 Introduction

Neural machine translation (NMT), as a new technology emerged from the field of deep learning, has improved the quality of automated machine translation into a significantly higher level compared to statistical machine translation (SMT) (Wu et al., 2016; Sennrich et al., 2017; Klein et al., 2017). State-of-the-art NMT models, like many other deep neural networks, typically contain multiple non-linear hidden layers. This makes them very expressive models, which is critical to successfully learning very complicated relationships between inputs and outputs (Devlin et al., 2014). However, as these large models have millions of parameters, they tend to over-fit during training phrases.

Dropout has recently been recognized as a very effective method to relieve over-fitting. Dropout was first proposed for feed-forward neural networks by Hinton et al. (2012). Dropout was then successfully applied to recurrent neural networks by Pham et al. (2014) and Zaremba et al. (2014). Dropout outperforms many traditional approaches to over-fitting, including early stop of training, introducing weight penalties of various kinds such as L1 and L2 regularization, decaying learning rate and so on.

Reported state-of-the-art NMT systems all adopt dropout during training phrases (Wu et al., 2016; Klein et al., 2017), but their paper have not provided many details about how dropout was applied. The optimal way to apply dropout to NMT models is non-trivial. Strictly speaking, NMT is an application of deep neural networks, so it can directly benefit from the advance of deep neural networks. However, two facts make NMT models stand out from normal deep neural networks. First, NMT models contain recurrent hidden layers in order to gain

the ability of operating on sequences. As contrast, deep neural networks in face recognition and phoneme recognition are feed-forward as they take separate inputs (Povey et al., 2011; Krizhevsky et al., 2012). Second, NMT models contain a novel attention mechanism to manage a memory of an input sequence, as this sequence can become quite long (Bahdanau et al., 2014). This attention mechanism involves calculations such as *weighted mean*, which is rarely used in normal deep neural networks.

As far as we know, there has been no focused study on how to apply dropout effectively to NMT models. This motivated the work presented in this paper, which aimed at finding out the optimal dropout scheme for training NMT models. Because the architecture of NMT models is complicated, we took an empirical approach. We trained many NMT models by verifying the subsets of connections that dropout were applied to, and using different dropout rates for different connections. We tried to find out the optimal dropout scheme through answering the following two questions,

- what part of NMT models should be applied dropout to during training phrases;
- how to correctly set the dropout rates for training NMT models.

The following of this paper is structured as: the section 2 reviews related works on dropout, then the section 3 describes the method that we adopted to search for optimal dropout scheme, after that the section 4 presents the results of the experiments, and in the end the section 5 concludes this paper with a description on our future work.

2 Related Works

Hinton et al. (2012) and Srivastava et al. (2014) proposed the method of dropout to relieve overfitting when training feed-forward neural networks. They worked with a variety of feed-forward neural networks each of which is established for a certain task. Different dropout schemes were applied to these neutral networks in order to obtain optimal results.

- MNIST is a standard toy data set of handwritten digits(LeCun et al., 2010). The best network had two layers of 8 195 rectified linear units. Dropout was applied to the input units with a rate $p = 0.2$, and the hidden units with $p = 0.5$.
- CIFAR-10 and CIFAR-100 are tiny natural images (Krizhevsky and Hinton, 2009); Street View House Numbers is a data set of images of house numbers collected by Google Street View (Netzer et al., 2011). The best network had three convolutional layers followed by two fully connected hidden layers. The output of the last fully-connected layer was fed to a softmax which produced a distribution over the class labels. All hidden units were rectified linear units. Each convolution layer was followed by a max-pooling layer. Dropout was applied to all layers including the input layer with $p = \{ 0.10, 0.25, 0.25, 0.50, 0.50, 0.50 \}$.
- ImageNet is a large collection of natural images (Deng et al., 2009). The best network had five convolutional layers followed by three fully connected hidden layers. The numbers of units in each layers were 253 440, 186 624, 64 896, 64 896, 43 264, 4 096, 4 096, and 1 000. The output of the last fully-connected layer was also fed to a softmax layer which produced a distribution over all class labels (Krizhevsky et al., 2012). Dropout was only applied to the first two fully connected hidden layers with $p = 0.50$. The reason might be that the network was very big . The author claimed that dropout roughly doubled the number of iterations required to converge. It can be inferred that applying dropout to all layers might not be feasible because of the time cost on training.

- TIMIT is a standard speech benchmark for clean speech recognition (Garofolo et al., 1993). The best network had six layers. Dropout was applied to the input layer with $p = 0.2$ and the hidden layers with $p = 0.5$.
- Alternative Splicing is a data set of RNA features for predicting alternative gene splicing (Xiong et al., 2011). The best network had two layers of 1024 hidden units. Dropout was applied to the input layer with $p = 0.2$ and the hidden layers with $p = 0.5$.

Their experiments were limited on feed-forward networks, which were much simpler than NMT networks. However, their experimental results suggest two useful heuristics about achieving good performance from dropout. First, dropout need to be applied to all layers of networks. Second, the optimal dropout rates for different type of layers such as input layers and hidden layers are different.

Zaremba et al. (2014) studied how to correctly apply dropout to recurrent neural networks such as long short-term memory (LSTM) units. They proposed that dropout should only be applied vertically, that is, be applied to non-recurrent connections. They argued that applying dropout to recurrent layers will amplify noise, as discussed by Bayer et al. (2013). They performed experiments on a variety of tasks, one of which was a machine translation task. Their NMT model contained no attention mechanism, which was like a recurrent language model trained on concatenations of source sentences and their translations (Sutskever et al., 2014). It had four layers of 1 000 LSTM units, three embedding layers (source language input embedding, target language input and output embedding) and a softmax layer. Dropout was applied to the connections between input-embedding-to-LSTM, LSTM-to-LSTM, LSTM-to-output-embedding [1] with $p = 0.2$. Their experiments were performed on a selected subset of the WMT 2014 English to French data set containing 340M French words and 304M English words Schwenk et al. (2011). Experimental results showed that dropout improved BLEU scores from 25.90 to 29.03, while still lost to a BLEU score of 33.30 achieved by a phrase-based SMT system named LIUM.

Wu et al. (2016) achieved a great improvement of translation quality through NMT when compared to their previous phrase-based production systems. They adopted a deep LSTM network that had eight encoder layers, eight decoder layers and a attention layer. They claimed that they adopted a dropout scheme similar to the method in Zaremba et al. (2014), but no further details were provided, especially on how dropout was applied to the attention layer.

Klein et al. (2017) released an NMT toolkit named OpenNMT. It implemented a network architecture similar with the one proposed by Luong et al. (2015). OpenNMT outperformed the SMT system of Moses (Koehn et al., 2007) and a few other NMT systems including GroundHog and Blocks in our pilot experiments. Therefore we took it as an important baseline in this paper. OpenNMT did not follow the dropout scheme of Zaremba et al. (2014) to apply dropout to all non-recurrent layers. Instead, OpenNMT applied dropout only to non-top encoding and decoding LSTM layers, and output hidden states(see section 3.2 for details). This dropout scheme seems arbitrary, but it did performed quit well in experiments. Solving this puzzle is one of the main motivations of this paper.

3 Methods

This section first presents the architecture of the NMT model that we adopted, which is one of state-of-the-art attention-based encoder-decoder NMT model. After that, this section analyzes that architecture and provides a list of connections in the architecture that are appropriate for applying dropout to. In the end, this section describes the method that we adopted to search for the optimal dropout scheme for training NMT models.

[1] According to the source code in `https://github.com/wojzaremba/lstm`.

3.1 Neural Machine Translation

The essence of a machine translation system is modeling the conditional probability of a translation given a source sentence. In encoder-decoder NMT models, it can be formalized using chain's rule as,

$$log\, p(\mathbf{y}|\mathbf{x}) = \sum_{j=1}^{m} log(p(y_j|y_1^{j-1}, \mathbf{s}) \tag{1}$$

where $\mathbf{x}$=$(x_1, \ldots, x_n, \langle \mathrm{EOS} \rangle)$ is a source sentence and $\langle \mathrm{EOS} \rangle$ is a end-of-sentence token; $\mathbf{y}$=$(y_1, \ldots, y_m)$ is a translation; $\mathbf{s}$ is an representation of $\mathbf{x}$ produced by the encoder. Note that $\mathbf{x}$ sometimes reverse its order to help the decoder to translate from the beginning of a source sentence.

In this paper, we adopted a compact stacking recurrent architecture as the encoder-decoder (illustrated by figure 1), which was proposed by Luong et al. (2015). This architecture assumes that equation 1 is factorized into a calculable form as,

$$\begin{aligned} log\, p(\mathbf{y}|\mathbf{x}) &= \sum_{j=1}^{m} log(p(y_j|H_o^{\langle j \rangle}) \\ &= \sum_{j=1}^{m} \log(\mathrm{softmax}_{y_j}(\tanh(W_o H_o^{\langle j \rangle} + B_o))) \qquad (2) \\ H_o^{\langle j \rangle} &= \mathcal{F}_{\mathrm{att}}(H_s, H_t^{\langle j \rangle}), \qquad (3) \end{aligned}$$

where H_s is a source-side hidden state produced by the top recurrent layer of the encoder; H_t is a target-side hidden state produced by the top recurrent layer of the decoder; H_o is a output hidden state produced by the attention model $\mathcal{F}_{\mathrm{att}}$; the superscript $^{\langle j \rangle}$ is a target-side timestamp; W_o and B_o are the matrix and bias of output embedding; and $\mathrm{softmax}_{y_j}$ means selecting the dimension from the output of the softmax which corresponds to y_j in one-hot encoding.

We adopted a global attention model (Luong et al., 2015) as $\mathcal{F}_{\mathrm{att}}$ (illustrated by figure 2). This attention model first calculates an alignment weight as,

$$\begin{aligned} a_{st}^{\langle ij \rangle} &= \mathrm{softmax}(\mathcal{F}_{\mathrm{a}}(H_s^{\langle i \rangle}, H_t^{\langle j \rangle})) \\ &= \frac{e^{\mathcal{F}_{\mathrm{a}}(H_s^{\langle i \rangle}, H_t^{\langle j \rangle})}}{\sum_{i=1}^{n} e^{\mathcal{F}_{\mathrm{a}}(H_s^{\langle i \rangle}, H_t^{\langle j \rangle})}}, \qquad (4) \\ \mathcal{F}_{\mathrm{a}}(H_s^{\langle i \rangle}, H_t^{\langle j \rangle}) &= H_s^{\langle i \rangle \top} W_{\mathrm{a}} H_t^{\langle j \rangle}, \qquad (5) \end{aligned}$$

where $\mathcal{F}_{\mathrm{a}}$ is a scoring function for alignment, which is composed of a linear mapping and a dot product; and W_{a} is a matrix for linearly mapping target-side hidden states into a space which is comparable to the source-side.

Then the attention model calculates translation contexts as,

$$\begin{aligned} C_s^{\langle j \rangle} &= \sum_{i=1}^{n} a_{st}^{\langle ij \rangle} H_s^{\langle i \rangle} \qquad (6) \\ C_{st}^{\langle j \rangle} &= [C_s; H_t^{\langle j \rangle}], \qquad (7) \end{aligned}$$

where $C_s^{\langle j \rangle}$ is a source-side context, and $C_{st}^{\langle j \rangle}$ is a context derived from both source and target sides through concatenating.

In the end, the attention model calculates an output hidden state as,

$$H_o^{\langle j \rangle} = W_c C_s t^{\langle j \rangle}, \tag{8}$$

where W_c is a matrix for linearly mapping a context into an output hidden state.

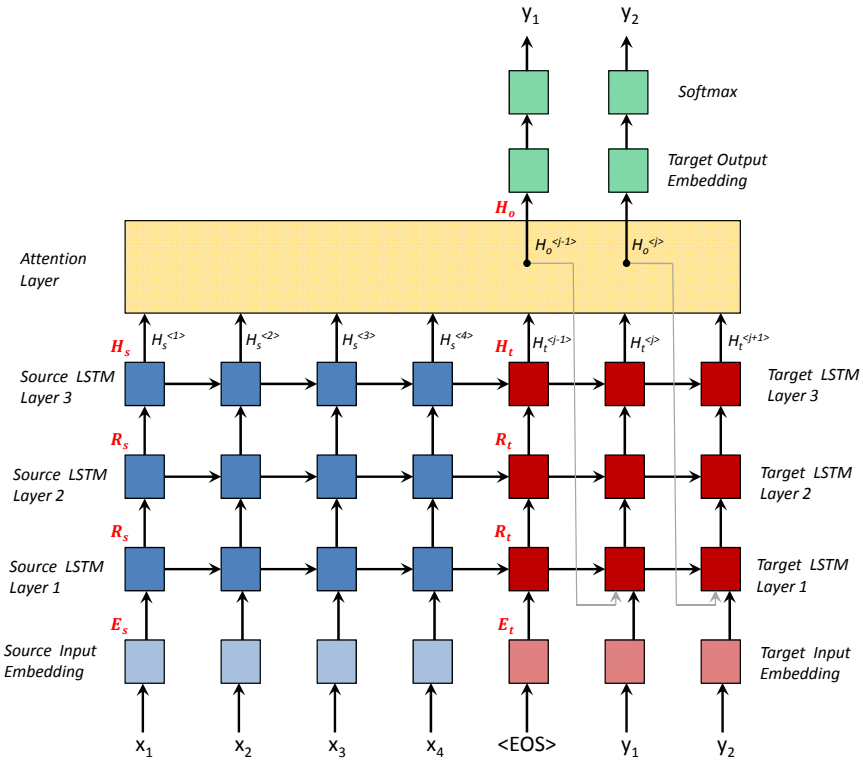

Figure 1: Network Architecture of Neural Machine Translation

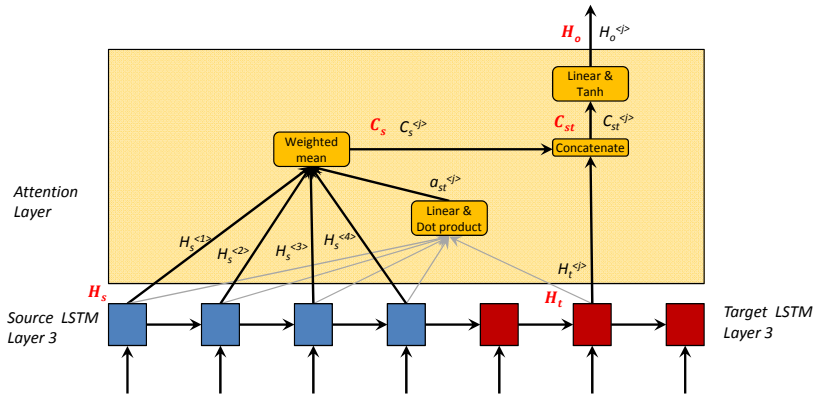

Figure 2: Illustration of Attention Model

3.2 Appropriate Connections for Dropout

Dropout should be applied vertically according to the previous study by Zaremba et al. (2014). This means that dropout should only be applied to non-recurrent connections. Therefore, there are nine connections in the NMT model that are appropriate for applying dropout to. The figures 1 and 2 annotate the corresponding variables with red colored symbols. For the sake of convenience, this paper views each connection in the NMT model as a variable. Applying dropout to a variable means applying dropout to those non-recurrent connections that transport the variable.

A detailed list of the nine variables appropriate for dropout in the NMT model is as follows,

- E_s input embedding of source-side;
- R_s hidden states of source-side non-top recurrent layers;
- H_s hidden states of source-side top recurrent layer;
- E_t input embedding of target-side;
- R_t hidden states of target-side non-top recurrent layers;
- H_t hidden states of target-side top recurrent layers;
- H_o hidden states of output;
- C_s translation context of source-side;
- C_{st} concatenate of source and target-side translation contexts.

The impact of applying dropout to these nine variables are not independent. Especially, the four variables involved in the attention model, including H_s, C_s, H_t and C_{st}, are closely related to each other. This is because the operators of *weighted mean* (the equation 6) and *concatenate* (the equation 7) reserve the effect of dropout. In other words, the dropout that is applied to their input will propagate onto their output.

Because of the complicated relations among these variables, the optimal way to apply dropout becomes a non-trivial task. For example, there are two choices if we want to make NMT model robust to source representations. One is to apply dropout to H_s, which will affect the calculation of alignment weight a_{st}, weighted mean C_s, and concatenated context C_st. The other one is to apply dropout to C_s, which will leave the calculation of alignment weight a_st untouched. It is quite difficult to predict the end-to-end performances of these two choices.

3.3 Search for Optimal Dropout Scheme

We decomposed the task of searching for optimal dropout scheme into two steps. The first step was to search for an optimal combination of variables for applying dropout to. The second step was to search for optimal dropout rates for each variable in the optimal combination.

Note that training NMT models was very time consuming, so exploring the full search space was impossible. Therefore we sometimes terminated searches early if one result was particularly good. We were aware that pruning search space might cause results not to be globally optimal. However, it made searching for optimal dropout scheme a feasible task.

3.3.1 Search for Optimal Combination of Variables

As described above, there are nine variables in the NMT model which are appropriate for dropout. The established toolkit of OpenNMT chooses to apply dropout to three of the nine variables, including R_s, R_t and H_o (Klein et al., 2017). This decision seems quite arbitrary.

Therefore, it was meaningful to find out which combination of these nine variables lead to good performance.

We took a heuristic greedy search to find the optimal combination of variables. We gradually increased the size of combinations. We started by applying dropout to only one variable. Then we tried applying dropout to two variables. After that we continued by applying dropout to three variables, and continued like this. In each stage we aimed to find the best combination of a given size. We generally tried adding variables to the promising combinations in the previous stage.

3.3.2 Search for Optimal Dropout Rates

Hinton et al. (2012) and Srivastava et al. (2014) showed that optimal dropout rates for different layers of feed-forward networks are different. Because the nine variables appropriate for dropout in the NMT model play different roles, they may have different optimal dropout rates. Therefore, we explored applying different dropout rates to the variables in the optimal combination, which was found in the first step.

We took a grid search to find optimal dropout rates . In each step, we tried increasing or decreasing the dropout rate of one variable by a fixed amount such as 0.05. We then chose the update on a variable which maximized the performance.

4 Experiments

This section first describes our experimental settings, then presents the results of searching for optimal combination of variables for applying dropout to, after that presents the results of searching for optimal dropout rates, and in the end compares our optimal dropout schemes with baselines.

4.1 Experimental Settings

Two corpora were used in our experiments (see the table 1). The first corpus was from the shared task of NIST Open Machine Translation 2006 Evaluation (OpenMT Chinese-to-English) [2]. We first removed the UN and the traditional Chinese data sets from the NIST-2006 constraint training resources. Then we performed word segmentation on the Chinese text using the Stanford word segmenter (Tseng et al., 2005), and performed tokenization on the English text using the scripts provided in (Koehn, 2005). The data sets of NIST Eval 2004 and 2005 were used as a development set. The data set of NIST Eval 2016 was used as a test set.

The second corpus was the Basic Travel Expression Corpus (BTEC) (Takezawa et al., 2002). We used the in-house English-to-Japanese corpus which contains about 463k sentences. We randomly selected 2 000 sentences as a development set, and selected another 2 000 sentences as a test set. The sentences left over were used as a training set. The English text was also tokenized using the scripts provided in (Koehn, 2005), and the Japanese text was segmented into words using the toolkit of Mecab (Kudo, 2005).

The wordpiece model was adopted to deal with rare words for NMT (Wu et al., 2016). This approach breaks all words, especially the rare ones, into subword units that are like to occur more often in a training corpus. Therefore, these rare words become translatable by NMT models. Byte Pair Encoding was adopted to train segmentation models (Gage, 1994). This method allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-length character sequences, making it very suitable for close-vocabulary systems like NMT (Sennrich et al., 2015). In this paper, we adopted a vocabulary size of 16k according to our pilot experiments.

[2] `http://www.itl.nist.gov/iad/mig//tests/mt/2006/`

Data Set	# Sentences	# Words		Vocabulary	
		Source	Target	Source	Target
Corpus: OpenMT Chinese-to-English					
Training	442,967	12,265,072	13,444,927	178,832	130,249
Development	2,679	72,869	87,369† (346,231‡)	NA	NA
Test	1,664	37,827	46,207† (193,214‡)	NA	NA
Corpus: BTEC English-to-Japanese					
Training	458,894	3,664,481	4,193,101	27.757	36,308
Development	2,000	16,148	18,451	NA	NA
Test	1,664	15,866	18,048	NA	NA

Table 1: Experimental Corpora. † the first reference; ‡ totally four references.

The phrase-based SMT toolkit of Moses was adopted as a baseline (Koehn et al., 2007). The Moses' models were trained in a conventional settings. The toolkit of SRILM was adopted to train 5-gram language models on target languages (Stolcke, 2002). The toolkit of Giza++ (Och, 2003) was adopted to perform word alignment. Then the training scripts provided by Moses were employed to build translation models. Then the systems were tuned with MERT on development sets (Och, 2003).

The NMT toolkit of OpenNMT was adopted as another baseline. OpenNMT outperformed Moses and a few other NMT systems including GroundHog and and Blocks in our pilot experiments. Therefore we took it as a baseline.

Different dropout schemes were tested using our C++ implementation of NMT, named CytonMT. The implementation utilizes NVIDIA's native libraries including CUDA, CUBLAS and CUDNN to gain efficiency on NVIDIA's GPUs. CytonMT adopts the network architecture proposed by Luong et al. (2015), which is similar with the one implemented by OpenNMT.

NMT models were trained with a similar setting as Luong et al. (2015). The stacking LSTM models had four layers of 1 024 cells, and 1 024-dimensional embedding. The parameters of neural networks were initialized in [-0.1, 0.1]. The parameters were trained with stochastic gradient descending algorithm. The gradient normalized gradient was re-scaled when it exceeded 5.

A simple adaptive learning rate schedule was employed to ensure that models with heavy dropout were fully trained. The training started with a learning rate of 1. If the perplexity on the development set did not decrease after an epoch, the learning rate started to decay by 0.5 per epoch. After that if the perplexity did not decrease in two continuous epochs, the training phrase was terminated. The maximum number of epochs was unlimited, while training usually finished around 20 epochs.

Translation performances of difference methods were measured by BLEU. BLEU was calculated on the lower-cased English words in the task of OpenMT Chinese-to-English, and was calculated on the Japanese characters in the task of BTEC English-to-Japanese.

4.2 Results of Searching for Optimal Combination of Variables

A group of experiments were performed following the method described in the section 3.3.1. The table 2 presents the results of the experiments. Main experiments were performed on the OpenMT corpus, and the BTEC corpus were used for confirming the findings.

The experiments were categorized into seven stages (separated by horizontal lines in the table), as the number of variables in the combination were gradually increased. In each stage, we aimed to find the best combination of given size (annotated by bold fonts in the table). We

generally tried adding variables to the promising combination in the previous stage. Because training NMT models s time consuming, we terminated the stage early if the best one was clear.

Two observations can be made from these experimental results. First, two special variables C_t and C_{st} are not suitable for dropout. They are called special because they are the output of *weighted mean* and *concatenate*, so they inherit the effect of dropout from the input. Experiments 21, 22, 29 and 30 show that the applying dropout to C_t or C_{st} leads to poorer performance than applying dropout to upstream variables H_s or H_t.

Second, among the seven remaining variables, the priority of applying dropout to each variable can be formulated as a chain,

$$R_t \succ R_s \succ H_o \succ E_s \succ H_s \succeq H_t \succeq E_t \tag{9}$$

where $\succ$ means context-ed superior, and $\succeq$ means context-ed superior or equal, with respects to dropout. $x_1 \succ \ldots \succ x_k \succ x_{k+1}$, means that given the context that $(x_1, \ldots, x_{k-1})$ have already been applied dropout to, applying dropout to x_k is superior to applying dropout to x_{k+1}. In other words, applying dropout to $(x_1, \ldots, x_{k-1}, x_k)$ outperforms applying dropout to $(x1 \ldots x_{k-1}, x_{k+1})$.

The priority chain of the equation 9 confirms that the OpenNMT's dropout scheme is an effective one, because R_t, R_s and H_o are the three top variables. Besides, the chain also suggests two other effective dropout schemes. The optimal-1 is to apply dropout to R_t, R_s, H_o and E_s. The experiment 17 shows that the cross entropy on the development set decreases by adding E_s into the OpenNMT's dropout scheme. The optimal-2 is to apply dropout to all the seven remaining variables. The experiments 23 – 27 show that adding any one or two from H_t, H_s and E_t into the optimal-1 brings little improvement, but adding all three variable into optimal-1 reduces the cross entropy.

4.3 Results of Searching for Optimal Dropout Rates

In this subsection, we aimed to refine the optimal dropout schemes found in the last subsection by using different dropout rates for each variable. We applied the grid search method described in the section 3.3.2. The table 3 and 4 presents the results of refining the optimal-2 on the OpenMT corpus and refining the optimal-1 on the BTEC corpus, respectively.

Unexpectedly, the experimental results on both corpora show that no changes on the dropout rates can improve the performances. Therefore, p = 0.3 is an optimal dropout rate for training the NMT model that we adopt.

4.4 Comparison with Baselines

In this section, we compared the optimal dropout schemes found in our study with baselines. Three baselines were employed. The first was Moses – one of the state-of-the-art phrase-based SMT systems. The second was the NMT toolkit of OpenNMT. The third was our implementation of CytonMT using the OpenNMT's dropout scheme. Among these three baseline, the third was the most accurate. The table 5 presents the results of the experiments.

Three observations can be made from the experimental results. First, on the OpenMT corpus, both the optimal-1 and the optimal-2 outperform the baselines (the experiments 1–5). In addition, the optimal-2 outperforms the optimal-1. This validates the effectiveness of optimal-1 and the optimal-2.

Second, on the BTEC corpus, the performances of the three dropout schemes are close. The different behaviors on the two corpora may be caused by the fact that the OpenMT corpus is more difficult than the BTEC corpus with respects to translation, as its sentences are longer and its vocabulary is larger. From the baseline to the optimal-1 and optimal-2, the strength of dropout gradually increases since more and more variables are being applied dropout to.

No.	Method: Apply Dropout to	Cross Entropy: Training	Cross Entropy: Development
	Corpus: OpenMT Chinese-to-English		
1	E_s	1.52	2.65
2	E_t	2.16	3.46
3	R_s	2.20	3.61
4	$\boldsymbol{R_t}$	1.59	**2.58**
5	H_s	1.66	2.85
6	H_t	1.63	2.62
7	H_o	2.33	3.51
8	C_s	2.24	3.56
9	C_{st}	1.74	2.70
10	R_tE_s	1.57	2.53
11	$\boldsymbol{R_tR_s}$	1.53	**2.46**
12	R_tH_t	1.72	2.53
13	R_tH_o	1.53	2.55
14	$R_tR_sE_s$	1.49	2.50
15	$E_tR_sH_t$	1.74	2.53
16	$\boldsymbol{R_tR_sH_o}$ $^\natural$	1.50	**2.41**
17	$\boldsymbol{R_tR_sH_oE_s}$ †	1.59	**2.36**
18	$R_tR_sH_oE_t$	1.70	2.40
19	$R_tR_sH_oH_s$	1.70	2.37
20	$R_tR_sH_oH_t$	1.68	2.40
21	$R_tR_sH_oC_s$	1.74	2.45
22	$R_tR_sH_oC_{st}$	1.70	2.43
23	$R_tR_sH_oE_sE_t$	1.59	2.37
24	$R_tR_sH_oE_sH_s$	1.65	2.36
25	$\boldsymbol{R_tR_sH_oE_sH_t}$	1.70	**2.36**
26	$R_tR_sH_oE_sH_tE_t$	1.72	2.37
27	$\boldsymbol{R_tR_sH_oE_sH_tH_s}$	1.79	**2.36**
28	$\boldsymbol{R_tR_sH_oE_sH_sH_sE_t}$ ‡	1.79	**2.33**
29	$R_tR_sH_oE_sE_tH_tC_s$	1.845	2.39
30	$R_tR_sH_oE_sE_tC_{st}$	1.834	2.41
	Corpus: BTEC English-to-Japanese		
31	$R_tR_sH_oE_s$ †	0.81	1.12
32	$\boldsymbol{R_tR_sH_oE_sH_tH_sE_t}$ ‡	0.83	**1.11**

Table 2: Results of Applying Dropout to Different Combinations of Variables. The Dropout rate is $p = 0.3$. $^\natural$ dropout scheme of the toolkit OpenMT; † the optimal-1; ‡ the optimal-2.

No.	Dropout Rate							Cross Entropy	
	R_t	R_s	H_o	E_s	H_t	H_S	E_t	Training	Development
Corpus: OpenMT Chinese-to-English									
Ref.	0.30	0.30	0.30	0.30	0.30	0.30	0.30	1.79	**2.33**
1	**0.35**	0.30	0.30	0.30	0.30	0.30	0.30	1.79	2.38
2	**0.25**	0.30	0.30	0.30	0.30	0.30	0.30	1.96	2.43
3	0.30	**0.35**	0.30	0.30	0.30	0.30	0.30	1.85	2.36
4	0.30	**0.25**	0.30	0.30	0.30	0.30	0.30	1.77	2.38
5	0.30	0.30	**0.35**	0.30	0.30	0.30	0.30	1.73	2.35
6	0.30	0.30	**0.25**	0.30	0.30	0.30	0.30	1.77	2.35
7	0.30	0.30	0.30	**0.35**	0.30	0.30	0.30	1.80	2.37
8	0.30	0.30	0.30	**0.25**	0.30	0.30	0.30	1.74	2.36
9	0.30	0.30	0.30	0.30	**0.35**	0.30	0.30	1.81	2.34
10	0.30	0.30	0.30	0.30	**0.25**	0.30	0.30	1.79	2.34
11	0.30	0.30	0.30	0.30	0.30	**0.35**	0.30	1.81	2.37
12	0.30	0.30	0.30	0.30	0.30	**0.25**	0.30	1.78	2.34
13	0.30	0.30	0.30	0.30	0.30	0.30	**0.35**	1.89	2.41
14	0.30	0.30	0.30	0.30	0.30	0.30	**0.25**	1.81	2.37

Table 3: Results of Using Different Dropout Rates on the optimal-2.

No.	Dropout Rate				Cross Entropy	
	R_t	R_s	H_o	E_s	Training	Development
Corpus: BTEC English-to-Japanese						
Ref.	0.30	0.30	0.30	0.30	0.81	**1.12**
9	**0.35**	0.30	0.30	0.30	0.73	1.13
10	**0.25**	0.30	0.30	0.30	0.73	1.13
11	0.30	**0.35**	0.30	0.30	0.82	1.12
12	0.30	**0.25**	0.30	0.30	0.77	1.12
13	0.30	0.30	**0.35**	0.30	0.73	1.13
14	0.30	0.30	**0.25**	0.30	0.76	1.12
15	0.30	0.30	0.30	**0.35**	0.80	1.13
16	0.30	0.30	0.30	**0.25**	0.79	1.12

Table 4: Results of Using Different Dropout Rates on the optimal-1.

No.	System	Dropout Scheme	Cross Entropy		BLEU	
			Train.	Dev.	Dev.	Test
Corpora: OpenMT Chinese-to-English						
1	Moses	NA	NA	NA	32.12	31.11
2	OpenNMT	baseline$^\natural$	1.62	2.42	39.96	39.11
3	CytonMT	baseline$^\natural$	1.50	2.41	40.07	39.21
4	CytonMT	optimal-1†	1.59	2.36	40.39	39.38
5	CytonMT	optimal-2‡	1.79	**2.33**	40.35	**39.89**
Corpora: BTEC English-to-Japanese						
6	Moses	NA	NA	NA	52.09	50.77
7	OpenNMT	baseline$^\natural$	0.63	1.15	52.35	52.38
8	CytonMT	baseline$^\natural$	0.81	1.12	52.49	**52.46**
9	CytonMT	optimal-1†	0.81	1.12	52.58	52.44
10	CytonMT	optimal-2‡	0.83	**1.11**	52.63	52.33

Table 5: Comparison with Baseline Methods. $^\natural$ baseline: OpenNMT's method, applying dropout to R_t, R_s and H_o. † optimal-1: applying dropout to R_t, R_s, H_o and E_s. ‡ optimal-2: applying dropout to all variables of R_t, R_s, H_o, E_s, H_t, E_t and H_s but exclude C_s and C_{st}. The drop rate is fixed as p = 0.3 .

Therefore, for easy translation tasks, the baseline or the optimal-1 is sufficient; while for difficult tasks, the optimal-2 is recommended.

Third, all the NMT systems trained with dropout clearly outperformed the SMT system. This indicates that the baseline dropout scheme is effective. This also confirms the description in the section 1 of this paper.

5 Conclusion

In this paper, we performed an empirical study on dropout scheme for training NMT models. We started the study by analyzing the architecture of NMT models, and found out the appropriate variables for applying dropout to. We then run two groups of experiments to find out the optimal combination of these variables and the optimal dropout rate.

Two main questions raised in the introduction can be answered through our study. The first question is what part of NMT models should be applied dropout. The priority of the variables in NMT models is

$$R_t \succ R_s \succ H_o \succ E_s \succ H_s \succeq H_t \succeq E_t.$$

This chain suggests some effective dropout schemes, including the OpenNMT's scheme, the optimal-1, and the optimal-2 (section 4.2). Note that heavy dropout scheme will increase the required number of epochs in training phase. If a translation task is difficult, and training time is sufficient, the optimal-2 is recommended (section 4.4). We empirically find that training NMT models using OpenNMT's dropout scheme usually converges within the 13 epochs [3] which is quite efficient, while using the optimal-2 usually requires around 20 epochs.

The second question is how to correctly set dropout rate for training NMT models. It is found that the dropout rate p = 0.3 is optimal for the NMT model that we adopt (section 4.3).

In the future, we plan to explore two topics related to dropout for NMT, including max-norm regularization (Srivastava et al., 2014) and drop connections (Wan et al., 2013).

[3] the default setting of the toolkit OpenNMT

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Bayer, J., Osendorfer, C., Korhammer, D., Chen, N., Urban, S., and van der Smagt, P. (2013). On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R. M., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1993). DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n*, 93.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, volume 5, pages 79–86.

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 the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kudo, T. (2005). Mecab: Yet another part-of-speech and morphological analyzer. *http://mecab. sourceforge. net/*.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Schwenk, H., Lambert, P., Barrault, L., Servan, C., Afli, H., Abdul-Rauf, S., and Shah, K. (2011). LIUM's SMT machine translation systems for WMT 2011. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 464–469. Association for Computational Linguistics.

Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., L"aubli, S., Miceli Barone, A. V., Mokry, J., and Nadejde, M. (2017). 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, Valencia, Spain. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stolcke, A. (2002). SRILM - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Takezawa, T., Sumita, E., Sugaya, F., Yamamoto, H., and Yamamoto, S. (2002). Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *LREC*, pages 147–152.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random field word segmenter. In *Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing*, volume 171. Jeju Island, Korea.

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066.

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*.

Xiong, H. Y., Barash, Y., and Frey, B. J. (2011). Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics*, 27(18):2554–2562.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.